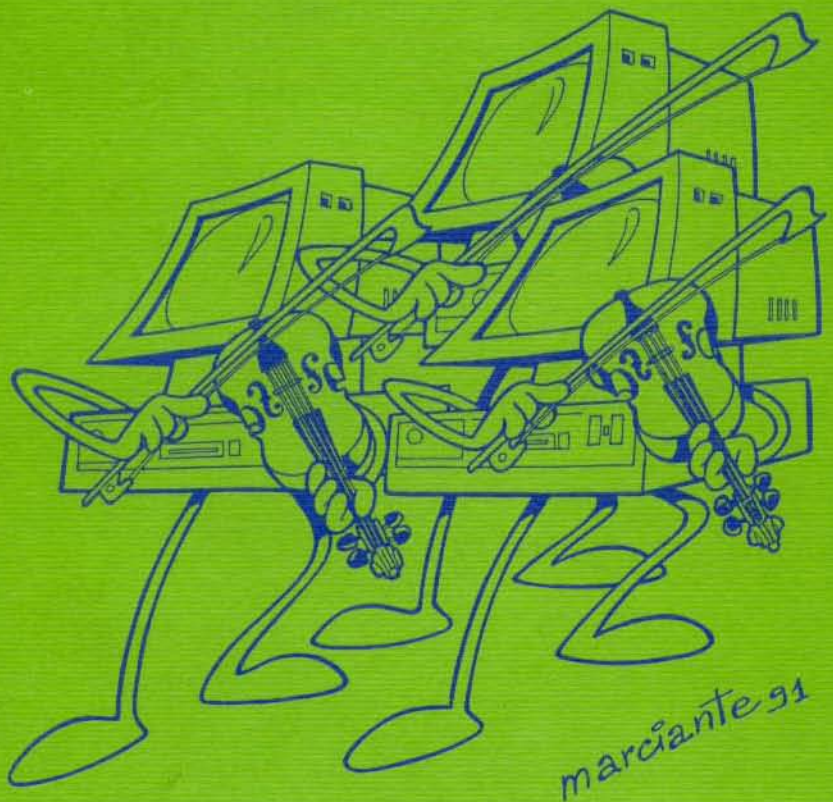
IX COLLOQUIO DI INFORMATICA MUSICALE

GENOVA
13 - 16 NOVEMBRE 1991

AIMI
Associazione di Informatica Musicale Italiana

DIST
Università di Genova
Laboratorio di Informatica Musicale



IX CIM

IX COLLOQUIUM ON MUSICAL INFORMATICS
IX COLLOQUIO DI INFORMATICA MUSICALE

AIMI - Associazione di Informatica Musicale Italiana
DIST - Università di Genova, Laboratorio di Informatica Musicale

Genova, 13-16 Novembre 1991

Sessioni Scientifiche:
Palazzo Tursi, Salone di rappresentanza, Via Garibaldi, 9 - Genova
Concerti:
Teatro Carlo Felice, Auditorium, Genova

PROCEEDINGS
ATTI

Edited by
A cura di

Antonio Camurri
(informatica)

Corrado Canepa
(musica)

Comitato scientifico:

Lelio Camilleri (Conservatorio di Firenze)
Antonio Camurri (DIST, Università di Genova)
Gianni De Poli (DEI-CSC, Università di Padova)
Giuseppe Di Giugno (IRIS, Frosinone)
Goffredo Haus (LIM-DSI, Università di Milano)
Aldo Piccialli (Università di Napoli)
Renato Zaccaria (DIST - Università di Genova)

Comitato musicale:

Mario Baroni (Università di Bologna)
Nicola Bernardini (Tempo Reale, Firenze)
Roberto Doati (Conservatorio di Cagliari)
Angelo Paccagnini (LIM-DSI Università di Milano)
Nicola Sani (RAI, Roma)
Sylvian Sapir (IRIS, Frosinone)
Alvise Vidolin (Conservatorio di Venezia)

Comitato organizzatore:

Antonio Camurri; Corrado Canepa; Federico Ermirio; Paolo Podesta`; Giuliano Palmieri; Maurizio Vento; Renato Zaccaria

Il Colloquio e` promosso da

AIMI - Associazione di Informatica Musicale Italiana

DIST - Dipartimento di Informatica, Sistemistica e Telematica dell'Università di Genova, Laboratorio di Informatica Musicale

con il patrocinio di

Comune di Genova, Assessorato alle Istituzioni e Attività Culturali

E. A. Teatro Comunale Dell'Opera di Genova

GOG - Giovine Orchestra Genovese

IEEE Computer Society, North Italy Section

Regione Liguria, Settore Beni e Attività Culturali

Hanno contribuito:

- *all'organizzazione*

DIST, Università di Genova

E. A. Teatro Comunale dell'Opera di Genova

Goethe-Institut Genua

GOG - Giovine Orchestra Genovese

Ministero Affari Esteri, Direzione Generale Relazioni Culturali

Musiquarium, Genova

SILICOMP Italia

The British Council

- *ai concerti*

ORTI SAULI, Galleria d'Arte in Genova

- *alla stampa del presente volume*

Comune di Genova, Ufficio Stampa e Pubbliche Relazioni

DIST, Università di Genova

GOG - Giovine Orchestra Genovese

Regione Liguria, Servizio Promozione Culturale

SUN Microsystems Italia

PROGRAMME

November, 13:

15.00: Welcoming Address

16.00 - 18.00: Presentation and Demo IRCAM
Cort Lippe, Zack Settel, Miller Puckette and Eric Lindemann
IRCAM, Paris
The IRCAM Musical Workstation: A Prototyping and Production Tool for Real-Time Computer Music
Rocco Parisi, clarinetto

21.00: Concerto di apertura / Welcome Concert
Quartetto di Asti (Silletti, Nuti, Reggio, Bovio), chitarre
Bruna Pannella, Elisa Soldatini, clavicembali
Rino Vernizzi, fagotto
Musiche di *Ermirio, Canepa, Palmieri, Graziani*

November, 14:

9.30 - 10.15: Invited Paper
Marc Leman
IPEM, University of Ghent - Editor of *Interface*
Tonal Context by Pattern Integration over Time

10.30 - 11.30: Neural Networks

R.Bresin, G.De Poli, A.Vidolin
CSC-DEI, Università di Padova
Reti neurali per il controllo delle deviazioni temporali nell'esecuzione musicale

M.Graziani, P.Zollo
CSC, Università di Padova - Università di Bologna
Classificazione timbrica per sintetizzatori MIDI mediante rete neurale

M.Graziani, P.Zollo
CSC, Università di Padova - Università di Bologna
Due reti con memoria a breve termine per la generazione di sequenze armoniche e di patterns ritmici in tempo reale

11.45 - 13.30: Demo/Poster/Video

M.Graziani
CSC - Università di Padova
Interazione di computer music e computer graphics in esecuzioni dal vivo

R.Bianchini, M.Gabrieli, S.Petrarca, G.Piazza
RAI, Divisione Televideo, Roma
Hypermusic

R.Bresin, G. De Poli, G. Torelli
CSC-DEI, Università di Padova
Applicazione delle reti neurali alla classificazione dei registri dell'organo a canne

R.Musto
Reminiscenze, videotape

L.Taiuti
10 videoclip di musica contemporanea, videotape

15.00 - 15.45: Invited Paper

Alan Marsden
Queen's University of Belfast
Musical Abstractions and Composers' Practice

15.55 - 17.15: Musicology

A.Provaglio, S.Sciarrino, A.Vidolin, P.Zavagna
CSC, Università di Padova.
"Perseo e Andromeda": opera lirica per mezzosoprano, tenore, baritono, basso e sistemi informatici di Salvatore Sciarrino. Composizione, realizzazione ed esecuzione

G.Cospito, V.Simmarano
Conservatorio "B.Marcello", Venezia
"Strumentazione" di musica per strumenti acustici con suoni sintetici: un esempio applicativo, Intercomunicazione per violoncello e pianoforte di B.A.Zimmermann

L.Camilleri, D.Bencini, M.Ignelzi
Divisione Musicologica del CNUCE/CNR - Conservatorio "L.Cherubini", Firenze
Analisi paradigmatica di repertori monofonici

A. Di Scipio
CSC, Università di Padova.
Caos deterministico, composizione e sintesi del suono

17.30 - 18.30: Demo/Poster

A.Piccialli, P.Basile, M.Sala, P.Simone, S.Vergara

Dip. Scienze Fisiche, Università di Napoli.
Una workstation per l'elaborazione dei segnali musicali e della voce

M.Giri
Composizione tramite automi cellulari

L.M.Del Duca, M.Marani, G.Bertini
Università La Sapienza, Roma - Leonardo Spa - IEI, Pisa
SUONO 2000 - Una workstation per l'educazione al suono

21.00: Concerto IX Colloquio di Informatica Musicale

Hans Van Dijck, Pietro Mianiti, viola
Musiche di *Ceccarelli, Doati, Di Scipio, Wolman, Sargenti, Cospito, Molino*

November, 15:

9.00 - 10.20: Digital Signal Processing (I)

F.Scalcon
DEI-CSC, Università di Padova.
A new contribution to noise analysis of the interpolating sinusoidal digital oscillator

G.Borin, G.De Poli, A.Sarti
DEI-CSC, Università di Padova
Una classe di algoritmi per eccitatori dinamici non lineari

G.B.Debiasi, G.Spagiari
DEI-CSC, Università di Padova
Metodi di analisi delle microvariazioni di ampiezza e frequenza dei suoni musicali e loro applicazione allo studio di un organo barocco

F.D'Agostino, I.Ortosecco
Cap Gemini SESA Italia - Dip. Scienze Fisiche Università di Napoli.
Physical models of woodwind instruments in a representation by means of wave digital filters

10.30 - 11.30: Digital Signal Processing (II)

A.Piccialli, P.Basile, M.Sala, P.Simone, S.Vergara
Dip. Scienze Fisiche, Università di Napoli
Analysis/Synthesis system based on a perceptive model

G.Evangelista
ACEL, Dip. Scienze Fisiche, Università di Napoli
Time-scale representations of musical sounds

B.Fagarazzi, M.Santinato, M.Sebastiani
CSC, Università di Padova

Codifica di eventi sonori e dei relativi parametri formali con il metodo dell'I.F.S.

11.45 - 12.45: Studio Report

G.Di Giugno, J.P.Armand et al.
IRIS, Frosinone
Studio Report IRIS

L. Tarabella, G. Bertini
Studio Report del Reparto d'Informatica Musicale del CNUCE/CNR di Pisa

A.De Vitis, M.Lupone, A.Pellecchia
CRM, Centro Ricerche Musicali
CRM: from the FLY 10 to the FLY 30 system

14.30 - 15.15: Invited Paper

Mitch Harris, Alan Smaill, Geraint Wiggins
Department of Artificial Intelligence, University of Edinburgh
Representing music symbolically

15.15 - 17.05: "The Intelligent Musical Workstation"

Consiglio Nazionale delle Ricerche, Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 7, Linea di Ricerca Coordinata C4: MUSIC

A.Camurri, G.Haus
DIST, Università di Genova - LIM-DSI Università di Milano
Architettura e ambienti operativi della Stazione di Lavoro Musicale Intelligente

G.Haus, A.Sametti
LIM-DSI Università di Milano
Un sistema per la sintesi di partiture MIDI mediante esecuzione di modelli formali

A.Ballista, E.Casali, J.Chareyron, G.Haus
LIM-DSI Università di Milano
Un ambiente integrato per la elaborazione musicale MIDI e la sintesi del suono mediante moduli DSP

A.Camurri, C.Canepa, M.Frixione, C.Innocenti, C.Massucco, R.Zaccaria
DIST, Università di Genova
HARP: un ambiente ad alto livello per l'ausilio alla composizione

U.Bertelli, C.Bima, A.Camurri, L.Cattaneo, P.Jacono, P.Podestà, R.Zaccaria
DIST, Università di Genova
Progetto SOUL: un sensore acustico adattivo per il riconoscimento di sorgenti sonore

17.05 - 19.00 Assemblea AIMI / AIMI Meeting

21.00: Concerto IX Colloquio di Informatica Musicale

Metella Pettazzi, arpa; Guido Boselli, violoncello; Loredana Marcolin, pianoforte
Musiche di *Bertoni/Serotti, Lippe, Galante, Dal Farra, Cospito/Simmarano*

November, 16:

9.00 - 9.45: Invited Paper

Stephen T. Pope
CCRMA Stanford University, The Nomad Group, Editor of the *Computer Music Journal*
The Interim DynaPiano: An Integrated Tool and Instrument for Composers

10.00 - 11.00: Computer Music Systems

L.Camilleri, F.Giomi, P.Graziani, L.Taggi
Divisione Musicologica CNUCE/CNR, Conservatorio "L.Cherubini", Firenze
Istituto di Ricerca sulle Onde Elettromagnetiche del CNR
Informatica musicale e non-vedenti: una stazione di lavoro su personal computer

G.Nottoli
ORLA Spa
ORION: un sistema per la sintesi e l'elaborazione del suono integrato in un singolo chip

A.Molino
CSC, Università di Padova
Il progetto "Live": un sintetizzatore virtuale e il suo utilizzo in concerto

11.00 - 11.45: Invited Paper

Christoph Lischka
GMD St. Augustin
On Music Making

11.45 - 12.25: Artificial Intelligence and Cognitive Science

U.Seifert
Università di Amburgo
The schema concept - A critical review of its development and current use in cognitive science and research on music perception

S.Girardi, G.Tisato
Civica Scuola d'Arte Drammatica "P.Grassi", Milano - CSC Università di Padova
Perceptual dimensions and acoustic correlations of laughter

22.00: Social Event

Sarà disponibile per tutta la durata del Colloquio una sala per ulteriori dimostrazioni delle installazioni presentate nelle sessioni demo/poster e video. Parte delle apparecchiature informatiche sono gentilmente fornite da Luschi Olivetti, Genova.

INDICE

Capitolo 8: dimostrazioni, video, poster

Parte 4: Concerti

A due anni dal *Workshop europeo su intelligenza artificiale e musica*, Genova ha l'occasione di ospitare la IX edizione del *Colloquio di Informatica Musicale*, evento di prestigiosa tradizione nel campo dell'informatica, della musica e del multimediale. Promosso dall'AIMI (Associazione Italiana di Informatica Musicale), il *Colloquio* è quest'anno organizzato dal Laboratorio di Informatica Musicale del DIST (Dipartimento di Informatica, Sistemistica e Telematica dell'Università di Genova). Si è voluto connotare questa edizione con uno spiccato contenuto scientifico e tecnologico, sottolineato da una significativa partecipazione internazionale, dalla qualità dei contributi e dal patrocinio della IEEE Computer Society, North Italy Section. Il *Colloquio* vuole rappresentare un punto di incontro, di scambio di esperienze fra le diverse realtà di questo particolare settore e costituisce inoltre un'occasione per la diffusione delle attività di ricerca e di produzione musicale, scientifica e industriale, sia a livello internazionale che locale.

Alcuni importanti temi scientifici, quali per esempio le reti neurali, l'intelligenza artificiale e le scienze cognitive, trovano ampio spazio in diversi capitoli del presente volume. In particolare, si è deciso di strutturare gli atti del convegno in quattro sezioni distinte. La prima, dedicata agli interventi dei relatori invitati, ha lo scopo di fornire punti di vista autorevoli e di scoprire le linee guida principali in alcuni tra i più rilevanti settori dell'informatica musicale. La seconda parte raccoglie i contributi scientifici: dall'elaborazione numerica di segnali, ai sistemi dedicati, all'intelligenza artificiale, agli algoritmi e ai metodi formali per la musicologia. Questa sezione comprende anche un capitolo sui più significativi risultati ottenuti dalle ricerche sulla "Stazione di Lavoro Musicale Intelligente", del *Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo*, del Consiglio Nazionale delle Ricerche, al quale lavorano due unità di ricerca: i laboratori di informatica musicale dell'Università di Milano (LIM-DSI) e del DIST di Genova. La terza sezione è dedicata alle comunicazioni, ai rapporti di attività, e alle descrizioni di sistemi per l'informatica musicale e il multimediale. Infine, la quarta e ultima parte, curata da Corrado Canepa, contiene le informazioni essenziali sui brani presentati ai concerti.

Siamo lieti che, grazie all'interessamento del Comune, della Regione, della GOG e dell'E. A. Teatro Comunale dell'Opera, il *Colloquio* sia stato ospitato presso il glorioso Teatro Carlo Felice, inaugurato proprio in questi giorni, e a Palazzo Tursi, antico palazzo patrizio nel cuore del centro storico cittadino, nonchè sede del Comune, simboli entrambi della rinascita della città di Genova.

Un sentito ringraziamento va al professor Roberto Negrini della IEEE Computer Society, North Italy Section; al professor Pier Paolo Puliafito, Direttore del DIST, per il supporto fornito alla manifestazione; al dottor Silvio Ferrari, Assessore alle Istituzioni e Attività Culturali del Comune di Genova; al dottor Josef Gerighausen, Direttore del Goethe Institut Genua, che tra i

primi ha aderito alla manifestazione con entusiasmo; a Mario Ingrassia della Giovine Orchestra Genovese, per la cortese disponibilità, il tempo e l'impegno profusi; a Carlo Bruzzo e Anna Sciacovelli della galleria d'arte Orti Sauli per gli utili suggerimenti; alla cortese e disponibile dottoressa Selo della Direzione generale relazioni culturali del Ministero degli affari esteri. Desideriamo inoltre ringraziare Enzo Marciante, e la signora Rasoli di Liguria Traduce per i preziosi consigli. Un sentito ringraziamento va ai colleghi dei comitati scientifico e musicale, e in particolare al presidente dell'AIMI Giovanni De Poli, ad Alvise Vidolin e Goffredo Haus. Un grazie infine agli autori, ai compositori e musicisti, agli studenti e allo staff del Laboratorio di informatica musicale (in particolare al dottor Frixione e all'ingegner Giuffrida), ai colleghi e agli amici del DIST, tra cui l'insostituibile Sergio, Roberto, Cinzia, Valentina, il personale dell'Amministrazione e della Segreteria, e a tutti quanti hanno contribuito alla riuscita della manifestazione.

Genova, 10 Ottobre 1991

Antonio Camurri, Maurizio Vento, Renato Zaccaria
per il comitato organizzatore

Presentazione

Giovanni De Poli
Presidente AIMI - Associazione di Informatica Musicale Italiana

L'AIMI (Associazione Italiana di Informatica Musicale) è stata fondata con lo scopo di promuovere e favorire lo sviluppo dell'informatica musicale e di diffonderne i risultati, sia nel mondo scientifico che musicale. L'AIMI costituisce il punto di incontro e cooperazione di due realtà generalmente autonome: il mondo musicale da una parte, l'ambiente della ricerca scientifica e tecnologica dall'altra. Nel corso di questi anni, l'AIMI ha riunito circa 600 soci in Italia ed ha patrocinato importanti manifestazioni quali la *International Computer Music Conference* di Venezia dell'82, la *International Conference on Musical Grammars and Computer Analysis* di Modena, i recenti *European Workshop on Artificial Intelligence and Music* di Genova e l'*International Workshop on Man-Machine Interaction in Live Performance* di Pisa. L'associazione è inoltre promotrice dei *Colloqui di Informatica Musicale* che si tengono ogni due anni, a partire dalla prima edizione di Pisa del 1976, presso le maggiori Università italiane.

Questo volume raccoglie i contributi scientifici e musicali presentati alla nona edizione del Colloquio di Informatica Musicale. Il Colloquio viene quest'anno organizzato dal Dipartimento di Informatica, Sistemistica e Telematica dell'Università di Genova, dove è attivo dal 1984 un laboratorio di informatica musicale, integrato con i preesistenti laboratori di intelligenza artificiale, di robotica avanzata ed il costituendo laboratorio multimediale. La scelta della sede lascia intravvedere la caratterizzazione prevalentemente scientifica di questa edizione della manifestazione.

L'obiettivo principale dei comitati scientifico e musicale è stato quello del raggiungimento della massima qualità e livello possibili dei lavori selezionati: a tale scopo, sono stati invitati cinque relatori provenienti dai maggiori centri di ricerca internazionali, è stata effettuata una severa selezione sui lavori pervenuti; infine, allo scopo di rendere maggiormente fruibili ai relatori stranieri ed in generale alla comunità internazionale i lavori presentati, si è deciso di adottare la lingua inglese accanto a quella italiana sia per le presentazioni che per gli articoli accettati. In particolare, ogni articolo in questo volume include almeno un riassunto in inglese. Dei circa 50 lavori sottoposti al comitato scientifico, ne sono stati accettati 16 come articoli estesi, 10 come articoli brevi e 10 come demo/poster. I contributi pervenuti al comitato musicale sono stati 26, di cui 12 accettati ed inclusi nei concerti serali del Colloquio.
Il livello raggiunto dalla manifestazione trova inoltre conferma nel patrocinio ottenuto dalla *IEEE COMPUTER SOCIETY, North Italy Section.*

Da tempo in Italia è in corso una notevolissima attività, sia in termini qualitativi che quantitativi, nel campo dell'informatica musicale. Questo ha determinato il fatto che in un Colloquio di Informatica Musicale non è più possibile presentare tutta l'attività in corso in Italia nelle sue svariate forme e sfumature. D'altra parte il prossimo avvicinarsi delle scadenze europee suggerisce di aprire ancor più i nostri incontri ad un respiro europeo, sia in termini di conoscere cosa avviene negli altri paesi che di confrontare la nostra attività con quella dei migliori ricercatori e musicisti europei. In altri termini, si vorrebbe cercare di avere anche a livello europeo un momento di incontro tipo il Colloquio

di Informatica Musicale. Per queste ragioni si cerca di andare nella direzione di qualificare ancora più il Colloquio di Informatica Musicale come momento di presentazione dei contributi più originali ed innovativi, e di sviluppare invece altri momenti per la presentazione di attività più applicative e di realizzazione, da inserire in un contesto più specializzato. E' con questo spirito che sono stati promossi i vari workshop, cercando cioè di individuare temi specifici ed emergenti o temi in cui vi sia un'attività diffusa ed un'esigenza di incontro e scambio di esperienze. In particolare si ritiene importante oltre agli incontri di tipo scientifico, promuovere anche incontri in cui si esaminano varie applicazioni e si affrontano i problemi pratici che si presentano ad utenti e musicisti.

Per finire, colgo l'occasione per ringraziare gli organizzatori, gli autori, i comitati e gli sponsor, che hanno contribuito, ciascuno nel suo ambito specifico, alla riuscita di questa manifestazione.

Presentation

Giovanni De Poli
President of AIMI - Italian Association of Musical Informatics

The Italian Association of Musical Informatics (AIMI) was founded with the aim of promoting and aiding the development of Musical Informatics and to spread its results, in both the scientific and the musical worlds. AIMI is the meeting and cooperation point of two generally separate realities: the music world from one side, the scientific research and industrial communities from the other. In the last decade, AIMI put together about 600 members in Italy and supported important events such as the *International Computer Music Conference* of Venice in 1982, the *International Conference on Musical Grammars and Computer Analysis* of Modena, the recent *European Workshop on Artificial Intelligence and Music* of Genova and the *International Workshop on Man-Machine Interaction in Live Performance* of Pisa. Furthermore, the association promotes the *Colloquia on Musical Informatics*, held each two years, starting from the first edition of Pisa in 1976, in the main Italian universities.

This volume collects the scientific and musical contributes presented at the ninth edition of the Colloquium of Musical Informatics. This year, the Colloquium is organized by the Department of Communication, Computer, and System Sciences (DIST) of the University of Genoa, in which a Laboratory on Musical Informatics is operating since 1984, strictly integrated with the pre-existent laboratories of Artificial Intelligence and Advanced Robotics, and, in the next future, with the DIST Media Lab, which is going to start its activities. The choice of this kind of organization let us foresee the mainly scientific characterization of this edition of the Colloquium.

The main goal of the musical and scientific committees has been the achievement of the maximum level and quality of the selected works: to this aim, five invited lecturers coming from the best international research institutes has been invited, and a severe selection among the submitted works has been made; besides, it has been decided to adopt the English and the Italian languages, both in the papers included in this volume and in the presentations, to make the works more enjoyable to the foreign lecturers and, in general, to the international community. In particular, each paper in this volume includes at least an extended abstract in English. 50 works were submitted to the scientific committee, and 16 have been accepted as long papers, 10 as short papers and about 10 as demo/poster. The contributes submitted to the music committee are 26, and 12 has been included in the program of the two evening concerts of the Colloquium.
The level of the Colloquium is also demonstrated by the support obtained from the *IEEE COMPUTER SOCIETY, North Italy Section.*

In Italy a growing activity is present in this field, both in qualitative and quantitative terms. This determined that in a Colloquium on Musical Informatics is not possible any more to present a full overview of all activities in Italy in all their forms and nuances. Nevertheless, the next European changes suggest us to open even more our events in an European context, both to know what is going on in other countries and to compare our activities with those of the best European researchers and

musicians. In other words, it would be important to have, also in a European context, a meeting event like the Colloquium on Musical Informatics. For these reasons, we go to the direction of qualifying even more the Colloquium on Musical Informatics as the event in which the most original and innovative contributes are presented, and to promote instead other moments for the presentation of more applicative or specialized activities, to be included in more dedicated contexts. This led us to the idea of promoting the different workshops, where specific and emergent research themes, characterized by a strong need of discussion and comparison among researchers, are faced. In particular, we believe it is important to promote, besides scientific events, also meetings in which different applications are examined, and the practical problems encountered by musicians and users in general are faced.

In conclusion, I take the occasion to thank the organizers, the authors, the committees, and the sponsors, which contributed, in their specific context, to the success of this event.

Parte 1: Relazioni su invito

TONAL CONTEXT BY PATTERN-INTEGRATION OVER TIME

Marc Leman

University of Ghent
Institute for Psychoacoustics and Electronic Music
Blandijnberg 2
B-9000 GHENT
Belgium

Keywords : Cognitive Musicology, Tonal semantics, Analogical Representation, Self-organisation, Recognition.

ABSTRACT

Music is characterized by a time-dependent information stream which gives rise to perceptual invariants (gestalts). A tonal context is considerd as a perceptual invariant of tonal music. It is determined by individual tones but whose meaning in turn is determined by the context. The dynamics of this process is discussed within the framework of a model of tonal semantics which predicts the emergence of perceptual schemata on the basis of self-organisation. It is now shown that these analogical representations may emerge in a very natural and ecological way : just by being exposed to tonal music. The concept of tonal context plays a central role in this theory. In this paper we provide an operational definition in terms of the integration of patterns over time and show its application in the field of automatic recognition.

1. Introduction.

The recognition of tonal centers has thus far been mainly associated with the analysis of scores. The system of Maxwell (1988) is an example of a rule-based (mini-) expert-system for harmonic analysis. The input is a score, the output is a set of symbols which reads as a harmonic analysis in text-book style.

The problem with this approach is that it bypasses the psychoacoustical foundations of harmony and tonality. Music theory is indeed a formalisation of musical intuition at a purely symbolic level. It does not formalise psychoacoustical prinicples such as *consonance* and *fusion* but assumes them only implicitly. Consequently, the mutual interplay between e.g. timbre and harmony is not accounted for.

Yet, we believe that concepts such as *consonance* and *fusion* are fundamental in our understanding of contemporary music as well as the historical development of tonal semantics. This idea could be

traced back to the work of v.Helmholtz (v.Helmholtz, 1863). He relied, however, on sensorical aspects of consonance and did not take into account aspects of *fusion* (Cfr. Terhardt, 1984). The latter concept was introduced by C. Stumpf in (Stumpf, 1890) and has been given an operational definition more recently by Terhardt (Terhardt, 1974, 1976). With the advent of the gestalt psychology and phenomenology in the first half of this century, the movement (known as "Systematische Musikwissenschaft") evolved more and more towards a metaphorical approach. Psychoacoustics, as part of psychofysics, became a science on its own while the systematic musicology moved towards the "human" sciences (with emphasis on esthetics and music theory, later also sociology).

However, with the advent of computers as well as the development of psychoacoustics, we are now in a better position to take up the old idea of a unified music science. Even the gestalt theory has gained a new impetus with the development of the socalled synergetic computers (Cfr. Haken & Stadler, 1990).

2. An application of cognitive musicology.

A complex tone such as a piano-tone consists of different simple tones that fuse together and determine the timbre as well. The combination of two such tones may be such that only the timbre is affected. E.g. when two tones are played on the piano with an interval of one octave the fused tone sounds more bright. But it may happen that we hear two tones "in harmony" sounding together. The synchronicity or asynchronicity of the two tones played may determine the difference between both. This clearly shows the diffuse boundary between timbre and harmony, the underlying prinicples of which are very essential for our understanding of musical cognition.

An other example of the effect that timbre can have on our perception is illustrated by the Shepard-tones. The spectral design is based on sinusoidal components at each octave of the frequency-domain. These components are filtered by a bell-shaped filter that enhances the fusion of the components. By this, the frequency-range perceived is reduced to one octave. A scale played over one octave ends up with a tone that has the same height as the one started from the lower octave. This circularity is the basis of socalled auditory illusions and paradoxes of musical perception[1]. In a musical context, Shepard-tones reinforce the effect of harmony by emphasising tonal *chroma,* (and reducing *height*) but the effect is somewhat reduced by sensations of *dissonance.* Consonance and fusion seem to compete with each other. A fugue of Bach played with Shepard-tones reduces the octave-spreading of tones to one octave and produces an effect which, apart from the perceived "one-dimensionality", is slightly more dissonant than when played with e.g. piano-tones. The dissonancy effect is mainly due to the fact that the interval of a seventh actually reduces to a second (because of the spreading of octave-components), and therefore falls within the critical zone of the ear where waves interfere (Zwicker & Fastl, 1990).

Given this background, we think that the automatic recognition of tonal centers from acoustical input is an appropriate test for the practical relevance of *cognitive musicology* in the field of music analysis and music history. At the same time the study is relevant for epistemological foundations and can be considered as a good application of artificial intelligence (some parallels with the

1. It is not my aim to go much deeper into this subject although the exploration of the principles that account for these effects is of utmost importance for a cognitive theory of music (See Deutsch et al., 1990, Terhardt, 1991).

recognition of speech are straightforward).

The essential processing mechanisms to be mastered deal with *consonance, fusion* and (as we shall see further) *self-organisation.* Cognitive musicology aims at developing a unified theory of musical perception and cognition in which these concepts play a fundamental role. We have argued (Leman, 1989) that this program can only be achieved by the development of an interdisciplinary approach based on psychology, musicology, philosophy, brain science and computer science [2].

In this paper we introduce a method that may contribute to this achievement by studying the way in which tonal context[3] may be established. We hope that some aspects of this approach could be useful for similar work in the field of timbral semantics as well. In many respects, however, this paper holds the middle between practice and theory. Some of the techniques developed have straightforward applications for computer-based (non-symbolic) music analysis, other aspects contribute mainly to the epistemological foundations of tonal semantics. We believe, however, that the mutual crossfertilisation between practice and more theoretical considerations is essential in the current stage of the work. Personally, we have the opinion that the epistemological background should be clarified before real applications in this field can set off. Hence, our presentation is mainly from the epistemological point of view.

We'll also talk a lot of computermodels, and simulatons. Experimentation with restricted models is believed to pose no fundamental epistemological problems as long as there is a straightforward way to extend the model towards more realistic proportions in the future. We think that this is possible with non-symbolic models, more than with symbolic models[4]. Therefore, during the presentation of the model we shall constantly keep an eye on how to extend the results obtained with such models.

3. A model of tonal semantics.

One of the problems in the automatic recognition of tonal context is how to deal with the information stream. Music is characterized by a time-dependent information stream which gives rise to the emergence of musical gestalts. A tonal context should be conceived as a musical gestalt. It can be interpreted as an perceptual *invariant* within which tones get their meaning. Our computer simulations show that a simple mechanism of pattern integration over time provides an initial good account for computing such an invariant. The study is related to a more general epistemological framework of the perception of tonal music, called "tonal semantics". Since the automatic recognition of tonal context should be considered as an application of this theory we cannot avoid going into some its details.

Tonal semantics deals with the question of how tones are related with each other and how tonal meaning arises from the interplay between *context* and tones. Recent research in cognitive psychology has come up with a method to quantify the perceptual schemata that underly tonal

2. In particular also studies in artificial intelligence. For an overview of the role of AI in music research, see Camurri (1990).
3. A tonal context is here understood as a tonal center or tonality. However, due to the music theoretical connotations of "tonal center" and "tonality" we prefer "tonal context".
4. For the distinction between symbolic and non-symbolic approaches in music research, as well as their epistemological implications, see Leman (in press).

semantics. This method is known as the *probe tone* technique (Krumhansl, 1990) : Listeners are asked to juge the similarity between the tonal context generated (e.g. by a cadence) and a tone. This experiment is repeated untill all chromatic tones of the octave have been juged. (In these experiments Shepard-tones are used in order to eliminate the influence of height.) Repeating this strategy for all tonalities one obtains patterns known as *key profiles*. They can be considered as distributed representations of tonal context. The set of key profiles can then be further processed with a mathematical technique such as multi-dimensional scaling analysis. By this method one obtains an insight into the *analogical* representation of the tonality centers. Multi-dimensional scaling of key profiles gives rise to the map depicted in Figure 1. The representation is *analogical* in the sense that the relationships between the objects on this map reflect the actual relationships between the objects in the real world [5]

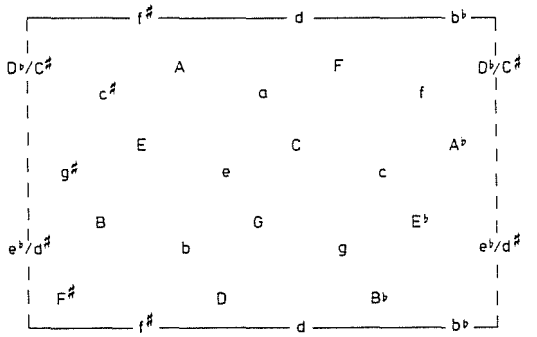


Figure .1. Multidimensional scaling solution (From : Krumhansl, 1990, p.46) of key profiles The points for the 24 major and minor keys are located on the surface of a torus.

This topology can also be obtained by the unsupervised classification of the key profiles in a neural network. This should be no surprise because self-organisation can be considered as a more powerful technique for classification than multidimensional scaling. The self-organisation of the key profiles in a neural network suggest a straightforward (non-symbolic) memory implementation for the analogical structure.

But there is more. It can be shown by computersimulation that this perceptual schema develops by self-organisation in a very natural and ecological way : just by being exposed to tonal music! Much of the methods discussed in this paper have been developed to demonstrate this[6].

---

5. Shepard & Chipman (1970, p. 2) have characterised this kind of representation as *isomorphism of second order*. They say : "The crucial step consists in accepting that the isomorphism should be sought - not in the first-order relation between (a) an individual object, and (b) its corresponding internal representation - but in the second-order relation betzeen (a) the relations among alternative external objects, and (b) the relations among their corresponding internal representations. Thus, although the internal representation of a square need not itself be square, it should (whatever it is) at least have a closer functional relation to the internal representation for a rectangle than to that, say, for a green flash or the taste of persimmon. By the "functional relation" between two internal representations we refer, most fundamentally, to the tendency of a response that has been directly associated with one to be aroused, also, by the activation of the other."
6. The idea has first been proposed in Leman (1989, 1990, 1991a).

Apart from the research in cognitive psychology there is evidence that tonal information processing is the result of complex sensory and perceptual processing. Psychoacoustics has come up with a number of models for pitch perception which rely on the idea that the perception of pitch is mediated by mechanisms that give rise to residual tones. The human auditory system constantly fuses tonal components, and perceives the fundamental of a tone, even when this fundamental is missing in the spectrum. In other words, the brain extracts the fundamental from the spectral residu[7].

	S-patroon :	1	0	0	0	1	0	0	1	0	0	0	0
octaaf	1.00	DO	-	-	-	MI	-	-	SOL	-	-	-	-
kwint	0.50	SOL	-	-	-	-	DO	-	-	-	MI	-	-
gr. terts	0.33	MI	-	-	SOL	-	-	-	-	DO	-	-	-
kl. septiem	0.25	-	-	DO	-	-	-	MI	-	-	SOL	-	-
gr. secunde	0.20	-	-	MI	-	-	SOL	-	-	-	-	DO	-
kl. terts	0.10	-	MI	-	-	SOL	-	-	-	-	DO	-	-
	R-patroon :	1.83	0.10	0.45	0.33	1.10	0.70	0.25	1.00	0.33	0.85	0.20	0.00
	O-patroon :	0.51	0.03	0.12	0.09	0.30	0.19	0.07	0.28	0.09	0.24	0.06	0.00
	th-klassen :	DO	DO#	RE	MIb	MI	FA	FA#	SOL	LAb	LA	SIb	SI

Figure 2. A simplified earmodel (From : Leman, 1991b).
The S-pattern (see text) is specified above, the R-pattern and O-pattern below. The contribution of a S-component to the subharmonic tone chroma (or R-component) is expressed in terms of the weights of the subharmonic intervals. The S-component DO contributes to the R-component DO because it is the second harmonic. The weight is therefore heigher than the contribution of the same S-component DO to the R-component FA, for which the DO is only the third harmonic. The weights are accumulated.

The "missing" fundamental we hear is called the *residu* tone because it results from a residu of the complete spectrum. It is beyond the scope of this paper to go too deep into the mechanisms by which residual pitch is produced. Some models assume a mechanism based on temporal information, while other models rely on pattern recognition of spectral information. Still other models combine both.

The model of Terhardt (1974) is well known in cognitive psychology. It can be simplified to a very simple mapping mechanism shown in Figure 2 and for that reason we choose it as the basis of our current experiments (Cfr. also Parncutt, 1988) [8]. Starting from a (simplified) spectral representation of chords, whose format is justified by the existence of Shepard-tones - their spectrum is reduced to

7. The Belgian telephone company (RTT) uses a bandwith of 300 to 3400 Hz for telephone communication. Although the fundamentals of voices are often lower than 300 Hz this poses no problems in hearing the pitch of the voice.
8. This model can of course be expanded by using a psychoacoustical earmodel.

one octave -, one can easily compute the evidence (sometimes also called probability) of the residual tones. This evidence (based on the acccumulation of overlapping subharmonics) gives an indication of the tone most probably heard. The spectral representation of a tone or chord is called a *S-pattern*, the residual pattern we call a *R-pattern*.

Our model predicts the emergence of perceptual schemata from S-patterns. The first level consists of a pitch extraction mechanism (a S-pattern is transformed into a R-pattern) and the second level consists of a unsupervised classification of the R-patterns. The second level is based on self-organisation, in particular we have adopted the Kohonen Feature Map (Kohonen , 1984).

In our first experiments we obtained good results with *time-independent patterns*, i.e., chords that are considered as time-independent objects (Leman, 1990; 1991a). The input is a set of S-patterns that represent the chords of tonal music. The output is a classification of these patterns in terms of a topology of neural functions in a neural network. An example is shown in Figure 3 and Figure 4. We used 115 different tonal chords that were input to the system as S-patterns. The picth-extraction mechanism transforms these patterns into R-patterns, which are given as input to the neural network. The network has a torus-structure (left and right, as well as high and low sides are connected) and the training was stopped after 400 training cycles. Afterwards the network was tested in order to get an idea of the internal structure of the analogical representation. It is believed that this internal structure should reflect the interrelations between the R-patterns. Figure 3 shows the topology of the network tested with key profiles (normalised). Figure 4 shows the response of the network to the tonal context of C. Recall that the key profiles are obtained by a method which is completely independent from the training patterns. Nevertheless, the results clearly show the appearance of the circle of fifths.

The topology shown in Figure 3 is called the *CN-topology* or : topology of characteristic neurons (Leman, 1991a). Characteristic neurons (CNs) are neurons that show the highest action in response to a pattern presented. They are assigned the label of the test-pattern (here : key profiles labeled as C, C# and so on). Inspection of the response-structure makes it possible to compare the knowledge-structure of the model with the data from cognitive psychology. The relationship is shown in Figure 5. The correlation between the data provided by Krumhansl (Krumhansl, 1990) and our computermodel is 0.99 for both the major and the minor context of C. This correspondence turned out to be very high but is somewhat lower for a network which has not this torus-structure. In a network with borders (studies in Leman, 1990, 1991a) the topology is not always preserving the correct distances (although the response structure is relevant) [9].

The relevance of the model is that it shows a straightforward way to build a bridge between psychoacoustics and cognitive musicology. In particular, this model shows how the organisation of concepts at a macro-level (whose structure is known from experimental research in cognitive psychology) may come into existence by the self-organisation of neuronal functions at a micro-level (starting from acoustical input).

To summarise : the model of tonal semantics combines a mechanism of tonal *fusion* and a mechanism of classification by *self-organisation*. In its extended form (on which we do not

9. For a detailed discussion of this we refer to Leman (1991b).

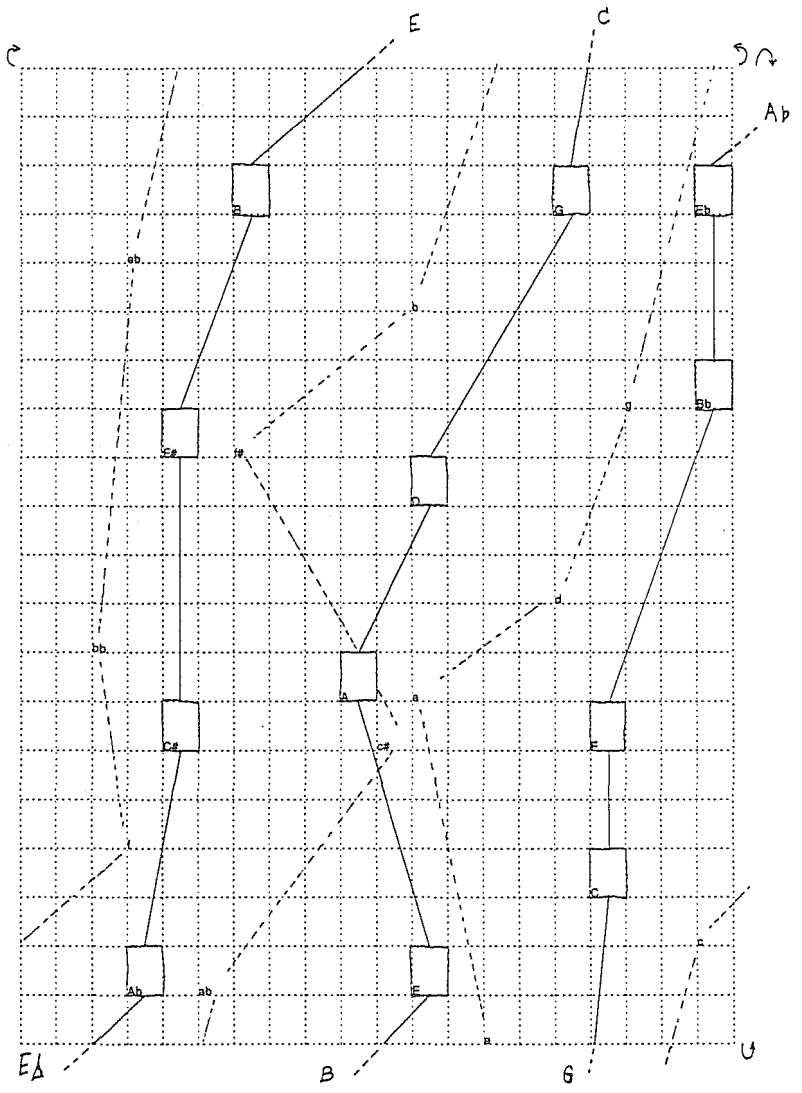


Figure 3. Self-organised topology of time-independent patterns (From : Leman, 1991b).
The network has a torus-structure (left and right sides as well as high and low are connected) with 400 neurons in a 20 x 20 raster. Each box represents a neuron. The neurons that give the highest response to a particular tonal context are labeled and their relationship is indicated. (Compare with Figure 1).

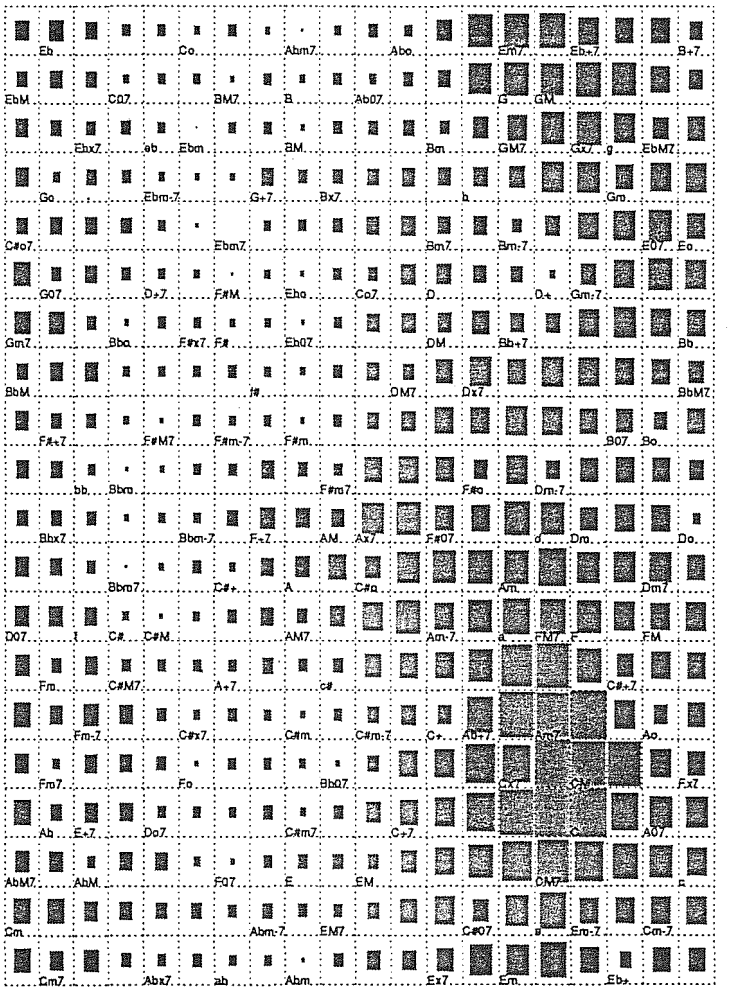


Figure 4. Respons the tonal context of C (From : Leman, 1991b).
The same network as in Figure 3 but the response to the tonal context of C is s shown. The CNs of tonal contexts as well as chords are shown.

elaborate here) it accounts for sensory (masking, pitch shift), as well as perceptual processing (pitch extraction) - e.g. when an earmodel such as Terhardt et al. (1982) is used - as well as cognitive processing (classification).

4. The complex dynamics of tonal semantics.

The model developed is limited at the level of the sensoric, perceptive and cognitive processing. One of its limitations pertain to the restricted capacity for giving a full account of the dynamics of

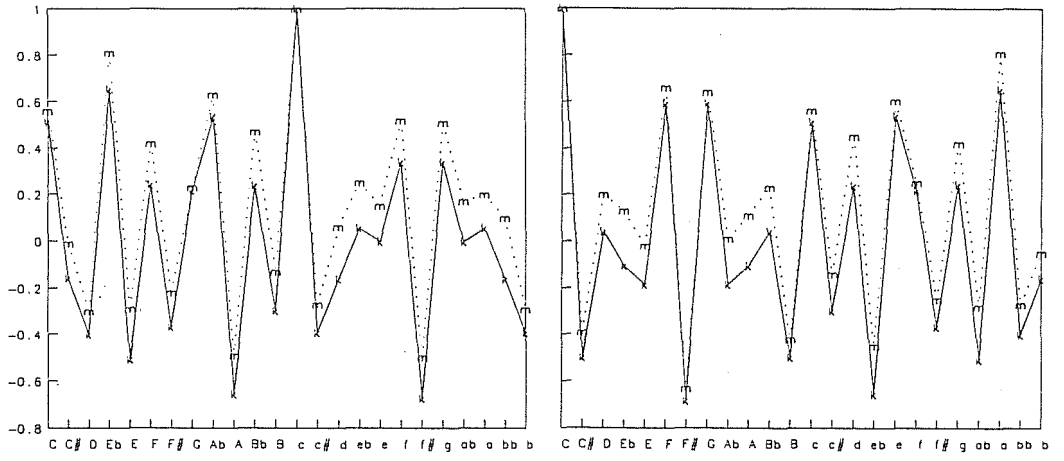


Figure 5. The similarity between tonal contexts (From : Leman, 1991b).
The figure compares the predictions of the model with psychological data of Krumhansl (1990). (a) All contexts are compared with C (DO-major). (b) All contexts are compared with c (DO-minor).

tonal semantics. We mean by this that the concept of self-organisation used does classification only in a very statical way. There is no account for dynamic alterability of percepts (concepts) int the time-domain. The current model creates spurious attractors without providing a dynamics for attractor transitions.

Self-organisation can indeed be considered as the dynamics of attractor-transitions. The concept of *synergetics* such as developed by Haken (Haken & Stadler (ed.), 1990) is much closer to this interpretation than it is to the interpretation of self-organisation as a learning process. However, we believe that some fundamental aspects of the dynamics can be simulated using several techniques of simplification, such as pattern-integration over time.

But before we proceed further in this direction we give a brief explicit formulation of the dynamics of tonal semantics in terms of dynamic systems. The paradigm is complicated from a dynamic point of view and such a characterisation might be helpful to understand our point of departure.

This is the metaphor : The perception of tonal music gives rise to the emergence of structured neuronal functions, called *perceptual schemata.* Once these schemata have been established, they support a dynamics of context-dependent similarities (Cfr. Figure 5)[10]. This should be conceived as follows : the musical information flow builds up a *context* against which new information is processed. The emergence of a context should be considered as the activation of a neural response pattern over time such that, once a sequence has been processed, a response pattern is activated in memory which influences the processing of further information. The memory establishes a

10. Actually, this dynamics is already supported during the formation of the perceptual schemata - which is actually what occurs during human learning-processes.

dynamics of contextual similarity such that tones get their *meaning* within the tonal context.

The difficult point to understand is, how the context can be determined by elements whose own meaning is determined by the context. We believe that the same situation occurs, at least to some extend, in natural language processing [11] : A sentence can indeed be considered as the context within which words, that constitute this sentence, get their proper meaning. Like tones, words have there *proper* meaning that builds up a context which then determines their *actual* meaning or *contextual* meaning.

The dynamics can also be understood in terms of attractor-dynamics. The tonal context can be considered as an attractor point to which nearby perceptions are forced. Formulated this way, we can say that the attractor can be disturbed by input. The fluctuations can ultimately cause a transition towards a new stable point of perception. A transition of a stable state happens for example with modulation : tones that are strange to the tonal center cause a fluctuation that leads to a transition of the system towards a new tonal center, which gives a quite different contextual meaning to the same tones.

It should be added also that this interpretation of tonal semantics is not so new. Fétis (Fétis, 1844) already interpreted harmonic processing is terms of tension and relaxation which comes very close to the concepts of complex dynamic systems (p. iij) :

> "[...] j'ai donc vu que toute l'harmonie réside dans ces nécessités alternatives : repos, tendance ou attraction, et résolution de ces tendances dans un repos nouveau."

5. Time-dependent information processing and invariants.

One of the central problems which we have not solved in the provious accounts of the theory, and which is relevant for the automatic recognition of tonal centers as well, is concerned with the way in which time-dependent information may give rise to the emergence of these perceptual schemata. This can in essence be reduced to the question of how a (tonal) *context* is builded up.

As mentioned, we believe that a tonal context might be interpreted as an perceptual *invariant*. According to Gibson, invariants are extracted from the spatio-temporal information flow. They correspond with constancy properties of the environment, which the brain can detect. The method we have used to obtain the invariants is by (temporal) *integration*. This assumes a mechanism by which the information flow is integrated in the activation of neurons. In other words, the memory is capable of holding information for a short time. This short-term memory seems to play an essential role for building up a mental context of the surrounding world.

6. Integration and neural synchronisation.

We found some support in the work of Pöppel (Cfr. Pöppel et al., 1990; Pöppel, 1989). According to Pöppel the temporal experience of humans can be described by a hierarchic classification of subjective temporal phenomena such as (i) *simultaneity, non-simultaneity* (asynchrony), (ii) *succession*, (iii) the subjective *present* (now) and (iv) *duration*.

---

11. For the rest we are very sceptical to compare natural language processing with musical processing, since musical processing is much more perceptualy based.

Simultaneity can be distinguished from non-symultaneity (asynchrony) if the temporal difference between the acoustical stimuli is at least 3 msec.

The perception of order (succession), on the other hand, requires 30 to 50 msec. This time-span corresponds with the minimal times that are required to make a conscious identification of some object. These values correspond with the standard deviation for differences in onset time of polyphonic music (Cfr. Rasch, 1979). If there would be no such differences, it would be much more difficult to distinguish the instruments. (Cfr. also our discussion higher).

The next step is the subjective sensation of the present. Events are not experienced in isolation. The single events are related to one another and create perceptive forms (gestalts) in which several events are combined. According to Pöppel this requires an integration of successive events into a perceptive form. His investigations in different domains (from analysis of spontaneous behavior, poetry, speech, ...) reveal a 3-second window of temporal integration, which corresponds roughly with what we found (see further). As suggested earlier, the tonal context may be conceived as a picture of the present, an invariant in the sense of Gibson, which determines the contextual meaning of perceptual objects at a smaller time-frame.

Finaly, duration is considered as an effect of structured memory representations. Information is somehow cumulatively stored and the stored information can be made available with respect to its duration.

One of the intriguing parts of this theory is the way in which the brain manages the information temporalily. Information is captured by different sense organs and processed by different information processing mechanisms that vary in speed (e.g. temporal fusion of visual and tactile stimuli are in the range of 20-30 msec and 10 msec, whereas for auditory stimuli this is 3-5 msec). How is it possible to combine the different sensations into the identification of one single object? According to Pöppel this can be explaned if one assumes that each stimulus activates an oscillating process in the brain such that neuronal oscillators are synchronized.

> "A visual, auditory or tactile stimulus causes periodic discharges in the stimulated neural networks. This creates a temporal framework, which enables events recorded by the various sensory systems to be correlated." (p. 85).

From a system theoretical point of view the correlated activity corresponds to a system state or attractor. It is suggest that the neural implementation of integration may be achieved by correlated activity of neurons. Recently, Reitboeck et al. (1990) have developed a computer model for feature linking via correlated neural activity which seems to be based on the same idea. They state (p. 115) that meaning can only be generated via associations and that the associate linking of sensory information with data structures in memory creates an interpretation of the physical stimuli in terms of previous experiences as well as in terms of inherited data structures. According to these authors, associations are established in the brain via temporal signal correlations, hence cognitive functions require correlated neural activity.

Pöppel's hypothesis is related to this in that the relaxation time for this correlation (necessary for identification) is in the order of 30-50 msec, while context (necessary for meaning) is established during a longer integration of 3 sec.

7. The temporal integration of MIDI-events for tonal context.

In this paragraph we describe a method for investigating the time window for integration. We do this on the basis of a program, designed for this purpose, in which the time window is variable. The best integration time is determined experimentally. The program, called *toonaard* actually consists of a series of smaller programs that work in a pipe on a Unix-system. It generates integrated patterns (called *I-patterns*) which can be compared with a manual analysis. On this basis the best time window is chosen.

In finding the time-span for integration we started from an idea that has been proposed by Schmuckler and Krumhansl (see Krumhansl, 1990). The algorihm is based on the computation of the correlation between a 12-dimensional vector, which represents the accumulated durations of the 12 pitchclasses, and all 24 key profiles. The output gives 24 correlations, the highest of which indicates the most prominent tonality.

The original formulation of this algorithm has been extended as follows :

- In order to avoid the manual input we start from MIDI-input.
- Instead of the accumulation of all durations we introduce a window whose time can be changed.
- Instead of starting from binairy vectors (the elements having values 1 or 0 depending on whether or not the chroma-component is present) we consider the introduction of an earmodel. As a first step this will be a simplified earmodel.

The input of MIDI-information is based on a program called *transcri* (Dannenberg, 1986). This program reads in MIDI-information from a keyboard and stores the information as symbolic code (called *Adagio-code*) in a file (called *gio*-file). Another program written by the same author (.I adagio), can play back the gio-file.

The gio-file is then transformed into a ''piano-roll'' representation. This is done with a progam called *sample*, which has been written for this purpose. *Sample* produces ''piano-roll'' samples of a gio-file at every n-hundreths of seconds. A ''piano-rol'' sample is simply a vector taken at a certain time where each element represents the key of keyboard. The values can represent the velocity (''loudness'') or can indicate whether the key is on or not (1 or 0). This depends on the options chosen. There is also an option for condensed output which gives 12-dimensional vector sample where each element represents a tone-chroma (pitch-class). Remember that such a condensed ''piano-roll'' vector is actually a S-pattern (see higher), hence, the spectrum is actually reduced to Shepard-tones.

A program *subhar* implements the simplified earmodel of Figure 2 and transforms 12-dim bin-vectors into 12-dim vectors that represent the tone pattern (the R-pattern). A program *opval* transforms R-patterns into normalised patterns, called O-patterns. The program *toonaard* produces temporal integration. In other words it transforms S-patterns, R-patterns or O-patterns into I-patterns and compares them with the key profiles stored in memory. The measure chosen is correlation.

8. Constraints for integration.

The type of window chosen was designed in order to fullfil three requirements for our notion of context :

1. The first requirement is that the context for a note should be dependent on its duration. Short durations should have a lower contextual (or integration) value than long durations since short durations are considered to be less determining for the formation of context than long durations.
2. Saturation can occur which means that after a certain duration the context value can reach a maximum.
3. When a note is off, then the system must forget it after some time.

Figure 6 shows the context value for one note with a duration of 100 steps (the duration of one step depends on the rate at which one produces the vectors with *sample).* The time-span w of the window has been set to 20 steps. This parameter can be taken as the unit of the contextual values. The context-value reaches (almost) the maximum after 100 steps. The value decreases in the next 100 steps.

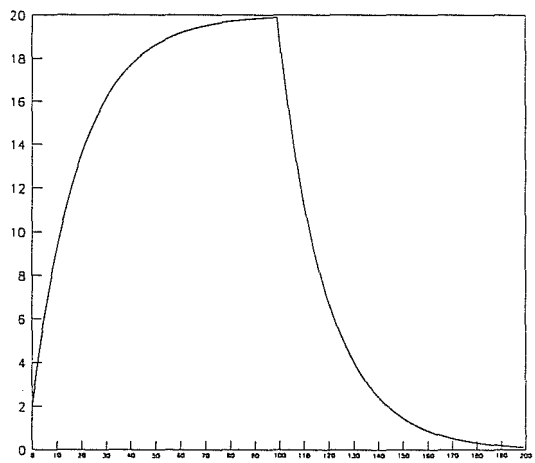


Figure 6. Integration curve (From : Leman, 1991b).

9. Determination of the integration time.

We compared three results : (a) integration based on piano-roll representation, (b) integration based on the ear-model *subhar,* (c) integration based on a normalisation of the ear-model patterns. In other words, the type of patterns used for integraton are based on S-patterns, R-patterns and O-patterns, respectively.

Figure 7 gives an overview of the development in tonal context for the first 45 measures of the Fantasia (KV 475) of W. A. Mozart. The horizontal axis is the time. The vertical axis is divided into 12 points each representing contextual values for a Shepard-tone (tone chorma). The piece has been recorded and processed according to the procedure described above and the analysis is based on the piano-roll representation (S-based I-patterns). The figure gives the evolution of the patterns (as a 12-dim vector) over time. Figure 8 gives an analysis of the same piece based on the integraton of R-patterns, while Figure 9 shows the analysis based on the integration of O-patterns.

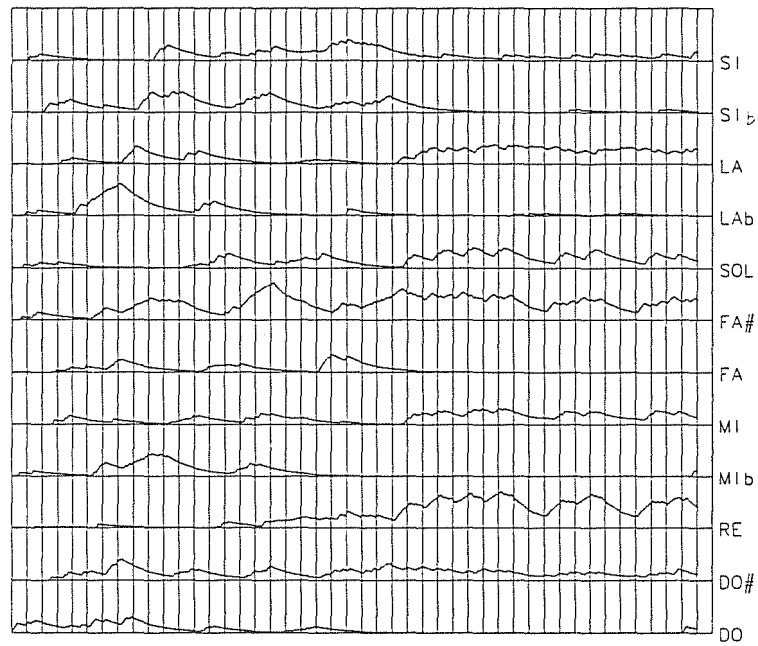


Figure 7. Development of tonal context (S-based I-patterns) (From : Leman, 1991b).

At each time-slice it is possible to compare the I-pattern (the tonal context at a certain time) with the key profiles so that the analysis can give a very detailed evolution of the most prominent tonal center. We compared the I-patterns at the end of evey measure with a manual analysis of the tonal centers by considering the most prominent correlations. If we only take the single most prominent correlation and compare the predicted context with the tonal center that is indicated manually, we find 75% correct answers for S-based I-patterns, 80% correct answers for R-based I-patterns and 88% correct answers for O-based I-patterns.

This shows :

- that the use of an earmodel gives better results than without earmodel,
- that normalised patterns gives better results than non-normalised patterns.

If we take instead of the single most prominent correlation the two most prominent correlations and count a correct answer if the tonal center found manually corresponds with one of the two, we find percentages of 84%, 88% and 95% for the three computer analyses respectively. By systematic variation of the window and application of this method for different pieces we found that the best integration time lies somewhat between 3 and 5 seconds. If the time-span is taken too short, then our analysis follows the chords, if the time-span is too large, then the analysis cannot follow the change of tonal centers. Up to now we only tested the model with the integration function of Figure 6. It is not excluded that better results can be obtained with other functions.

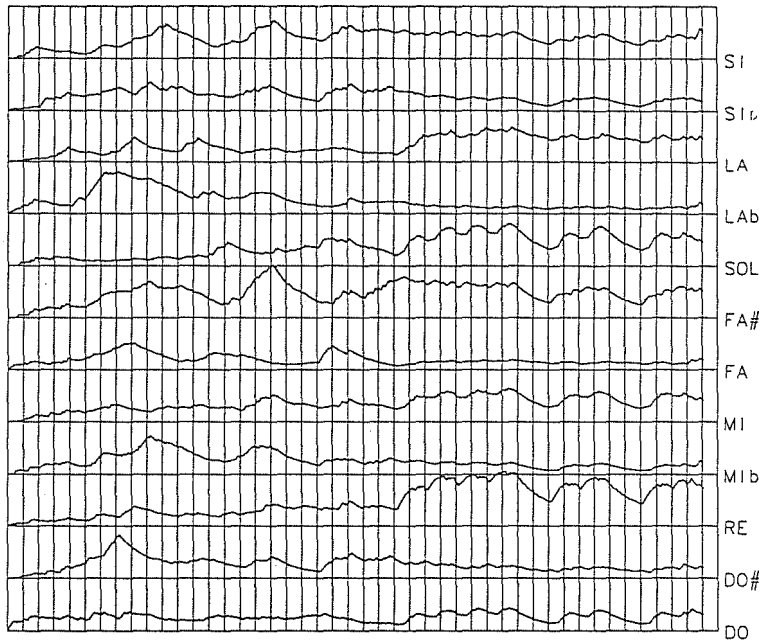


Figure 8. Development of tonal context (R-based I-patterns) (From : Leman, 1991b).

10. Testing the model with I-patterns.

The process of intergration is simulated by the integration of patterns over time. We trained our network with I-patterns to strenghten the hypothesis that perceptual schemata develop by a self-organising interaction with the musical environment.

We therefore took a piece of music from Fétis's *Traité de l' Harmonie,* in which the major and minor tonality are clearly established. We transposed this piece over all chromatic tones and computed the O-based I-patterns starting from tones with spectra simular to Shepard-tones. Afterwards we tested the netwerk with the patterns that we used in other studies to test the tonal centers. These patterns were computed by taking the mean value of chords in typical cadences.

Figure 10 shows the CN-topology of the tonal contexts for a torus-network trained with time-dependent patterns. Figure 11 compares the response-structure of the model with psychological data from Krumhansl. As can be noticed the model predicts these data but the correlations are less high than in the previous experiment (0.91 and 0.91 for the C major and C minor contexts, respectively).

On the whole this experiment in the recognition of tonal context from MIDI-input supports the hypothesis that 3 sec might be a reasonable time for the integration.

11. Conclusion.

The model of tonal semantics suggests a straighforward way for an application in field of tonal context recognition from acoustical input. The current earmodel can easily be exchanged for a psychoacoustical earmodel and instead of computing the correlations between the I-patterns and the

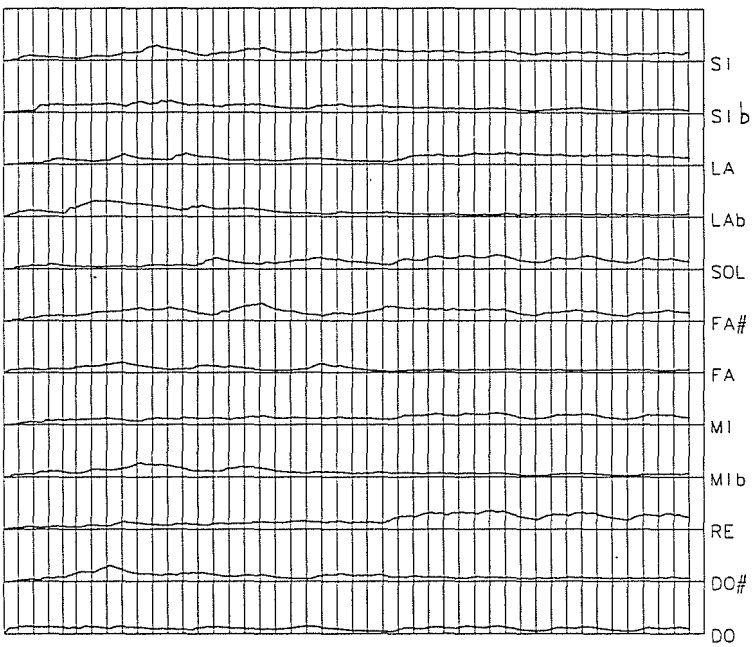


Figure 9. Development of tonal context (O-based I-patterns) (From : Leman, 1991b).

key profiles we can rely on the analogical representation of the tonal invariants in an neural network to see how the network responds to the I-patterns of acoustical input. The change of tonal centers is then represented as a trajectory on the map.

The relevance of this work is

- from an epistemological point of view : (a) Reduction of music theory to a cognitive theory of tonal semantics based on a unified theory of sensory, perceptual and cognitive processing of tones. (b) Elaboration of a method for the determination of the integration time for tonal context.
- from a practical point of view : (a) The recognition of tonal context from MIDI input. (b) Elaboration of a method to handle acoustical input.

We further believe that the methodology might provide a basis for a similar study in the field of timbral semantics.

ACKNOWLEDGEMENTS

We wish to thank H. Sabbe and the F.K.F.O (N.F.W.O.) for support.

REFERENCES

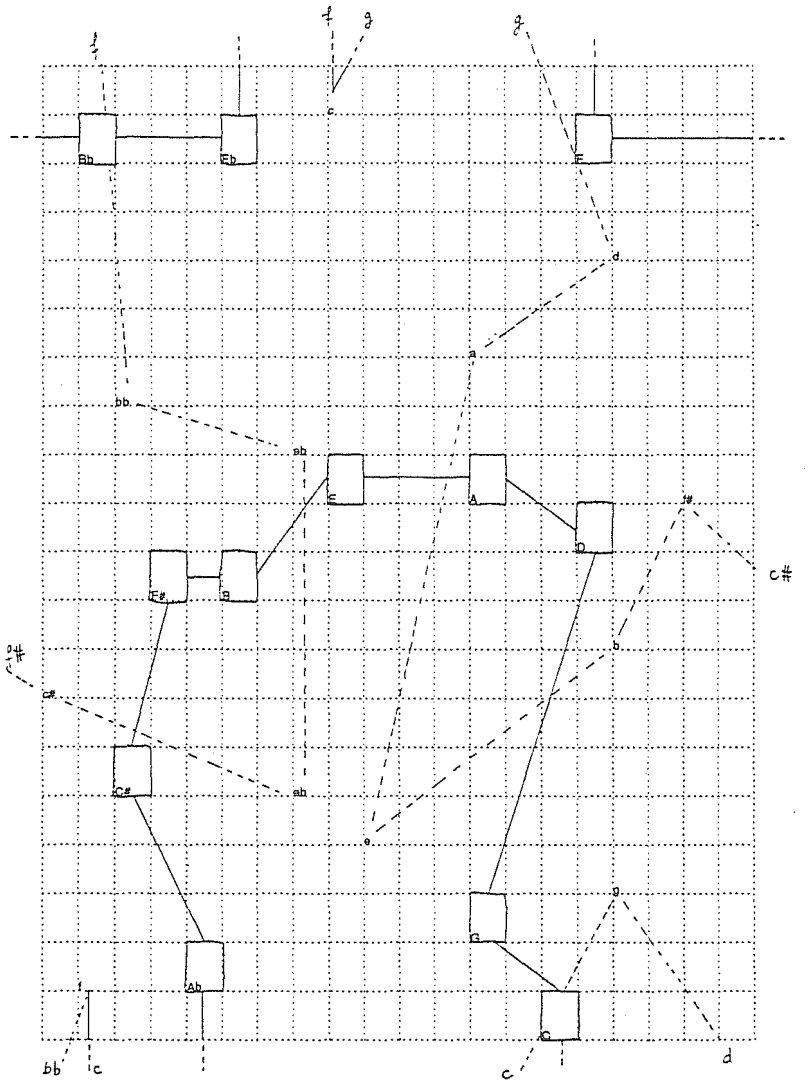


Figure 10. Self-organised topology of time-dependent patterns (From Leman, 1991b).

A. Camurri, "On the role of artificial intelligence in music research," **Interface**, vol. 19, no. 2-3, pp. 219-247, 1990.

R.B. Dannenberg, The CMU MIDI Toolkit, Carnegie Mellon University, Pittsburgh, 1986.

D. Deutsch, T. North, and L. Ray, "The tritone paradox : correlate with the listener's vocal range for speech," **Music Perception**, vol. 7, no. 4, pp. 371-384, 1990.

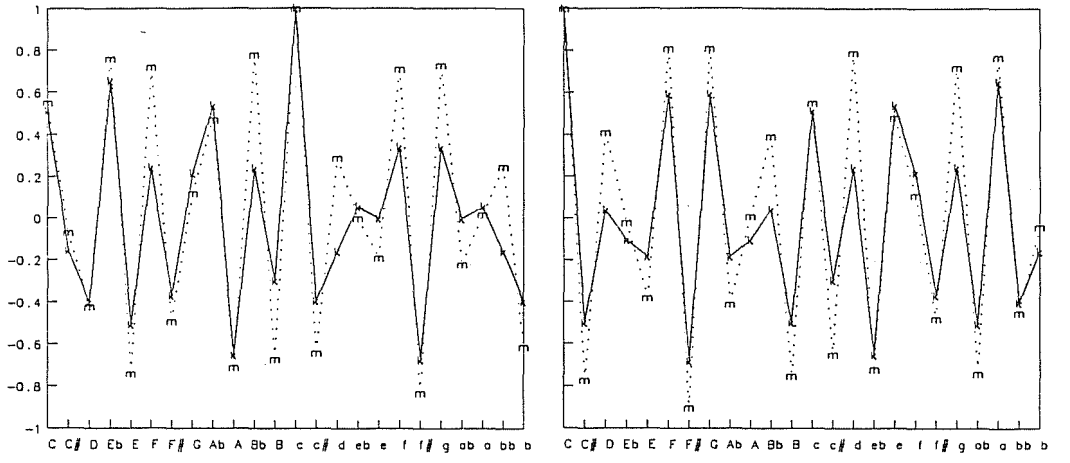


Figure 11. The similarity between tonal contexts (From : Leman, 1991b).

F.J. Fétis, **Traité complet de la théorie et de la pratique de l'harmonie**, Maurice Schlesinger, Paris, 1844.

H. Haken and M. Stadler (eds.), **Synergetics of cognition**, Springer-Verlag, Berlin, 1990.

H. v.Helmholtz, **Die Lehre von den Tonempfindungen als physiologische Grundlage für die Theorie der Musik**, Georg Olms Verlagsbuchhandlung, Hildesheim, 1863/1968.

T. Kohonen, **Self-organization and associative memory**, Springer-Verlag, Berlin, 1984.

C.L. Krumhansl, **Cognitive foundations of musical pitch**, Oxford University Press, New York, 1990.

M. Leman, "Introduction to models of musical communication and cognition," **Interface**, vol. 18, no. 1-2, pp. 3-7, 1989.

M. Leman, "Emergent properties of tonality functions by self-organization," **Interface**, vol. 19, no. 2-3, pp. 85-106, 1990.

M. Leman, "The ontogenesis of tonal semantics : results of a computer study," in **Music and connectionism**, P. Todd and G. Loy (eds.), The MIT Press, Cambridge, MA, 1991a.

M. Leman, Een model van tonale semantiek : aanzet tot een theorie van muzikale cognitie, University of Gent, doctoral dissertation, 1991b.

M. Leman, "Symbolic and subsymbolic description of music," in **Music processing**, ed. G. Haus (ed.), A-R Editions, in press.

H.J. Maxwell, "An expert system for harmonic analysis of tonal music ," in **Proceedings of the first workshop on artificial intelligence and music**, ed. Balaban et al. (eds.), AAAI-88, Minneapolis/St.Paul , 1988.

R. Parncutt, "Revision of Terhardt's psychoacoustical model of the roots of a musical chord," **Music perception**, vol. 6, no. 1, pp. 65-94, 1988.

R. Parncutt, **Harmony : a psychoacoustical approach**, Springer-Verlag, Berlin, 1989.

E. Pöppel, "The measurement of musicand the cerebral clock : a new theory," **Leonardo**, vol. 22, no. 1, pp. 83-89, 1989.

E. Pöppel, E. Ruhnau, K. Schill, and N. v.Steinbüchel, "A hypothesis concerning timing in the brain," in **Synergetics of cognition**, H. Haken and M. Stadler (eds.), Springer-Verlag, Berlin, 1990.

Rasch, R.A., "Synchronization in performed ensemble music," **Acustica**, vol. 43, pp. 121-131, 1979.

H.J. Reitboeck, R. Eckhorn, M. Arndt, and P. Dicke, "A model for feature linking via correlated neural activity," in **Synergetics of cognition**, ed. H. Haken and M. Stadler (eds.), Springer-Verlag, Berlin, 1990.

R.N. Shepard and S. Chipman, "Second-order isomorphism of internal representations : shapes of states," **Cognitive Psychology**, vol. 1, pp. 1-17, 1970.

C. Stumpf, **Tonpsychology II**, Verlag von S. Hirzel, Leipzig, 1890.

E. Terhardt, "Pitch, consonance, and harmony," **The Journal of the Acoustical Society of America**, vol. 55, no. 5, pp. 1061-1069, 1974.

E. Terhardt, "Ein psychoakustisch begründetes Konzept der Musikalischen Konsonanz," **Acustica**, vol. 36, pp. 122-137, 1976.

E. Terhardt, G. Stoll and M. Seewann, "Algorithm for extraction of pitch and pitch salience from complex tonal signals," **The Journal of the Acoustical Society of America**, vol. 71, no. 3, pp. 679-688, 1982.

E. Terhardt, "The concept of musical consonance : a link between music and psychoacoustics," **Music perception**, vol. 1, no. 3, pp. 276-295, 1984.

Terhardt, E., "Music perception and sensory information acquisition : relationships and low-level analogies," **Music Perception**, vol. 8, no. 3, pp. 217-239, 1991.

E. Zwicker and H. Fastl, **Psychoacoustics : facts and models**, Springer-Verlag, Berlin, 1990.

MUSICAL ABSTRACTIONS AND COMPOSERS' PRACTICE

Alan Marsden

School of Music, The Queen's University of Belfast, Belfast BT7 1NN, Northern Ireland
E-mail: A.Marsden@qub.ac.uk
Fax +44-232-247895

Keywords: Musical Abstractions, Musical Representation, Music Theory, Software Design

ABSTRACT

In designing musical software, whether the intended application is in composition or in some other area, it is necessary to decide on the nature of musical information to be represented. This inevitably involves some degree of abstraction. The purpose of this paper is to explore the basis on which this abstraction is made. In many cases abstraction is quite informal, being based on the introspections of composers or musical researchers, or traditional, being based particularly on past patterns of musical pedagogy. In the area of computing, however, the process of abstraction is more critical because once an extensive software system is built on one set of abstractions, one does not want to discover that the set is incomplete, inaccurate or simply wrong.
The intention here is to attempt abstraction on a more formal basis by examining composers' practice as shown in their works. The principles used are that where common elements exist, or where parsimonious descriptions of the music or composers' practice is possible, an appropriate abstraction is suggested. As illustration, two possible abstractions of rhythm are defined in a simple formal language, and representations of segments of music and of musical operations based on these abstractions are tested for parsimony.

Introduction: Canon

Computer systems and musicians have in common that they deal in abstractions. Yet, as argued in Marsden & Pople (in press) and demonstrated throughout that volume, relating the two is far from straightforward. The central question is "what nature of information must a computer system handle in order to perform some musical task?". In other words, what is the domain of the functions the computer system is to embody. The interim answer is of course that it depends on the particular task, but the most useful computer tools are those which are capable of a variety of tasks, some perhaps even unenvisaged by their designers. This paper does not aim to give even a partial answer to the question, but rather to explore a policy for arriving at an answer.

As an illustration of the significance of the nature of abstraction used, consider a computer system designed to perform the age-old musical task of realising a canon. To be more precise, imagine a pianist is given a single line of music and asked to play a strict two-part canon from it with a given time interval between the entry of the first and second parts — this is not an unreasonable task: if the pianist is unable to play it straight off, he or she could easily write out the necessary notation and play from that. Now we want to design a computer system to perform the same task, but we make no prescription

as to the nature of the representation used for the original line of music or the time interval. (A practical use for the system might be as an aid for composing canons, taking the donkey work out of the task.)

Let us first adopt a policy of as little abstraction as possible and use digital audio as our representation of music. (We might, for example, sing the original to the computer through an analogue-to-digital converter.) The task is now easy to specify (though it might take a lot of processing time!): if the time interval is N samples, the first N samples are reproduced exactly, thereafter the number representing the first sample is added to the number N+1 in the sequence, etc. until the final sample is reached, after which the last N samples are also reproduced unchanged. If any number in the resulting sequence is greater than the maximum allowed for a sound sample, then each number in the sequence will need to be scaled by the maximum divided by the greatest number in the sequence. Now the computer can play back its canon by sending the resulting sequence to a digital-to-analogue converter.

Our pianist, however, is quite likely, even without being asked, to play a canon at the octave rather than the unison in order to increase the separation of the parts (or simply to make it easier to play with two hands!), i.e. the second line will be a reproduction of the first *an octave higher or lower*. We know that an octave means a doubling in frequency, so we can reproduce our original sound sequence an octave higher by doubling the sample rate, but this also shortens the sequence by half. To avoid this we must use complex signal-processing techniques which have varying degrees of success and which considerably complicate the design of the system. The problem arises because the representation used does not abstract from the music separate domains of pitch (or frequency) and time.

Let us instead then adopt a policy of least resistance by making use of what is most readily available (by no means a bad policy), in this case MIDI. The original musical line could be played on a MIDI instrument, and the stream of MIDI data recorded by the computer, with each message being time-stamped by reference to the computer's internal clock (or by some other means). Now to produce the canon the computer can output the original sequence of MIDI data, preserving the time intervals between messages to preserve the rhythm, and also output a copy of that data delayed by the given time interval. To perform an octave transposition, the system must treat the MIDI data as more than simply a stream of bytes and be able to recognise those bytes which represent pitches so as to add or subtract 12 when making the copy. (A check should also be made that the resulting pitches are within the acceptable MIDI range.)

This is quite straightforward, but it cannot be guaranteed to be always successful because there might be occasions when the two lines of music coincide on the same pitch as, for example, in the fragment of canon in Figure 1. When the computer's output is played on a synthesiser, the middle C in bar 2 will be stopped after just a quaver's duration, instead of being held for a dotted crotchet as it should be (and as a pianist would do). The root cause of this is that MIDI is essentially a representation of the depression of keys (among other switching operations) on a music keyboard. Thus when the "note off" message representing the end of the middle C in the upper part arrives, the synthesiser interprets this as an instruction to perform the operation appropriate to a key being released, i.e. stopping the note. The fact that there have been two "note on" instructions is ignored because it is impossible to depress a key on a keyboard twice without releasing it in between.

One solution to this problem is to output the two voices through separate MIDI channels. (This will require a little more sophistication from the system's handling of its data because it must now also recognise those portions of the byte stream which represent channel numbers and change them

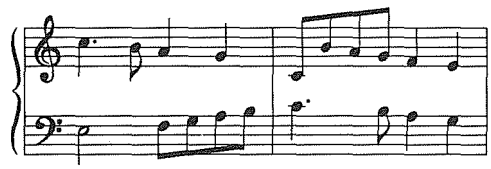


Figure 1. Fragment of canon with "collision" or parts

accordingly.) However, this is cheating a little: it is as if we have given the pianist two keyboards. If we are to mimic that part of the human pianist's operation which deals with coinciding pitches we must ensure that the unwanted "note off" messages are not sent. This is not actually difficult to do, but it does mean that the process is no longer one of simply copying the given sequence. Furthermore, whichever method we use to avoid the unwanted "note off" messages involves associating "note on" and "note off" messages for the same pitch — in other words we are no longer treating the data simply as key presses but as abstract compounds of "on" and "off", analogous to the musical concept of a "note", and there is not necessarily a one-to-one correspondence of notes with on-off pairs: musicians are quite happy with the idea that a single key press can in certain contexts stand for two notes.

A second problem with the MIDI representation is that simple delay and copying of the originally sequence will only be satisfactory if the original was played with a very strict tempo. Any fluctuations in the performance will mean that notes which a pianist would play together when realising the canon will not be synchronised in the computer's output. To determine correct synchronisation, the time intervals in the sequence must be treated in a categorical manner, and this turns out to be far from straightforward (see for example the research concerned with inferring metre from keyboard performances such as Lee, 1985).

The task is much simplified if we start with a representation of the original sequence in which (i) notes are explicitly represented, and (ii) temporal information is already categorical. We might, for example, represent a piece of music as a sequence of triples (onset time, pitch, duration), each representing a note. The pitch representation might be an integer representing a key on a keyboard, as in MIDI, and both onset times and durations might be multiples of the greatest common divisor of all the nominal durations in the piece. (A representation scheme something like this is examined below.) It is now a relatively straightforward task to write a computer program to copy the original line of music, making an octave transposition if necessary by adding or subtracting 12 to the pitch components, adding the given time interval for the canon to the onset times, and then interleaving the copy into the original so that onset times occur in increasing order.

The final step to allow the mimicking of most common canonic practice is to make a pitch representation which allows for easy transposition by intervals other than an octave, performing this transposition so as to remain "in key" for tonal music.

Policies for the Definition of Musical Abstractions

It might be assumed that a prudent policy when formulating musical abstractions is to use musical notation as a basis. This, after all, is where the preceding discussion of canons appears to be leading. Common practice notation isolates notes, associating each with a single symbol, and separates pitch (indicated by vertical positioning, possibly with modifications) from time (horizontal positioning, plus

different shapes to indicate different durations.) Furthermore, the notation of pitch facilitates tonal transposition. However, there are aspects of notation which if slavishly mimicked in a computational representation can cause difficulties in processing. For example, the strict use of barlines in musical notation, with apparently significant divisions such as key, time signature and tempo changes almost always coinciding with them, might suggest that the bar should be a significant unit of representation. However, if a line of music is represented as a sequence of bars, each a compound of notes, then operations which involve realigning sequences of notes relative to barlines, as for example when realising a canon in which the interval between the entry of the first and second voice is half a bar (not an uncommon occurrence), become more difficult because each bar must be decomposed into its sequence of notes and then new bars composed to reflect the new alignment. (Precisely this situation arises in the popular score-editing program "Finale": operations involving part-bars such as making an anacrusis or inserting a repeat sign part-way through a bar, are more difficult than they should be because the representation used is rigidly based on bars.) The root of the problem is that barlines serve a dual purpose in music notation. On the one hand they can represent something significant, viz. stress patterns. On the other hand they also serve the strictly non-musical purpose of providing a metric for the notation to aid the reader in locating notes etc. (the American choice of the term "measure" is telling here). (It is interesting in this regard that the designers of the draft ISO representation for music, SMDL, intend to include stress patterns as an element of the "logical" level of representation, but barlines as an element of the "visual" level (ISO, 1991).)

In fact many computational representations are based at least in part on music notation, and there is a certain logic to this — notation is the only concrete musical product other than sound which can provide a sure basis for representation. However, it is never represented simply as it is: a two dimensional arrangement of symbols. A considerable degree of interpretation of the arrangement precedes representation. For example, the occurrence of two note-symbols one above the other can be interpreted in three ways: if they are "on" the same staff (the use of ledger lines means that they need not be on the staff at all) they are interpreted as two simultaneous notes of different pitches; if they are on different staves within the same system they are interpreted as two simultaneous notes in different parts (unless the staves are part of a pair such as used for keyboard parts); if they are on staves of different systems, they are interpreted as two notes occurring one before the other.

Such interpretation is done according to a shared "language" for reading musical scores — perhaps this language might form a basis for computational representation. Informally this, again, is what frequently has been done in the past, but the central question is precisely "what are the elements and the grammar of this language?". (We will return to this later.) Some account of the language is contained of course in musical textbooks, but where these are not contradictory they are generally quite informal. A response (and again a not uncommon one) is to apply a policy of rationality to the material of musical textbooks, or other elements gleaned from musical pedagogy, analytical writings etc. The advantage of this policy is that a rational representation scheme is likely to facilitate computational processing. However, there are two dangers to this approach. Firstly, musical practice is not entirely (or at least not straightforwardly) rational. For example, on the basis that most intervals which are inversions of each other are in the same category of consonance or dissonance (e.g. 3rds and 6ths, 2nds and 7ths), and on the basis of small integer ratios of fundamental frequency, 4ths might rationally be considered to be consonant, but in most Western musical practice they are not. Secondly, there is a danger of getting carried away by the rationality and proposing things beyond the scope of general musical practice. To return once again to canons, for example, one might rationally suppose that, since there are canons by

rhythmic augmentation and diminution where the time intervals of a second part are expanded or contracted by a constant ratio in comparison with the corresponding intervals in the first part, a canon could be written with augmentation of pitch intervals also. However, I am not aware of a single case of this in actual practice.

An opposite policy might be an empirical one which recognises that the significant elements of musical notation, pedagogy, and indeed those handled by our hypothetical pianist realising the canon, are *cognitive* ones. So the experimental cognitive psychology of music might be regarded as furnishing a sound basis on which to build a computational representation. Indeed, experimental work does provide evidence for categorisation of pitch intervals and time intervals, and many other elements commonly regarded as musically significant (e.g. Burns & Ward, 1982; Deutsch, 1982; Sloboda & Parker, 1985), but findings are too imprecise and incomplete, and likely to remain so for some time, to provide a full basis for a computational representation. Furthermore some experimental work suggests that certain musical processes, which composers appear to have regarded as significant, judging from their writings and compositional practice, go unrecognised by listeners. This is often the case for retrogression for example. A representation which forbade retrogression would present difficulties for some canonic writing, let alone much 20th-century practice.

The Policy of Parsimony

If experimental psychology cannot (yet) furnish us with an account of the cognitive elements manipulated by composers (and other musicians) what about a careful scrutiny of their products? Might we not look at the "utterances" of the "language" referred to above and hope to find its elements by analysis? This is a common practice in linguistics and semiotics, and analogous to the analytical procedures proposed by Ruwet (1972) and machine learning proposed by Kippen & Bel (1989). In the Ruwet and Kippen & Bel cases, the basic procedure is to find recurrences: where some pattern of units occurs, an element of the "language" is created and associated with that particular pattern. What I want to propose here is a similar procedure, but to expand the notion of recurrences so that whenever two sequences can be construed to be each a well defined instantiation of some common "background" element, that element is used as part of the representation ("language"), together with the instantiation function.

The underlying principle here is one of parsimony. To return to simple recurrence, if we take the sequence ABCDEFABCDEFABCDEF we find three occurrences of the sequence ABCDEF, so this can become an element of our language. The sequence can now be represented as SSS:S→ABCDEF, where the colon indicates that what follows is an instantiation function, and the arrow means that every occurrence of the character preceding the arrow in the main sequence is replaced by the sequence of characters following the arrow. The representation is 12 characters long compared to the 18 of the original sequence, and therefore more parsimonious. (Even when we take into account the increase in the size of the alphabet used in the representation (9 distinct symbols compared with 6 in the original) the representation is more parsimonious: 38.0 bits ($12\log_2 9$) compared with 46.5 bits ($18\log_2 6$).) Furthermore most of the second representation is taken up with the instantiation function, so if we are to represent a longer sequence or additional sequences containing further occurrences of the sequence ABCDEF we will make even greater savings.

A "background recurrence" occurs in the sequence ACDFEBBDEGFCDFGBAE, which could be represented as in Figure 2. (Here we must allow for variables in instantiation functions — the symbols

```
ASBSDS:S→21274;
x0→xx;x1→xx+;
2→1+;3→2+;4→3+;5→4+;6→5+;7→6+;
A+→B;B+→C;C+→D;D+→E;E+→F;F+→G;G+→A
```

Figure 2. Representation of the sequence ACDFEBBDEGFCDFGBAE

x or y can be unified with any one symbol in the work string; a semicolon indicates the end of an instantiation function.) The recurrence is of the sequence S, but this is a sequence of *intervals* (represented here by numbers). The representation is less parsimonious than the original sequence, but increases in the length of the original sequence will not require significant increases in the size of the representation, provided it continues to use the same recurrent pattern. The representation constitutes about 398 bits of information ($92\log_2 20$), equivalent to a sequence of about 142 symbols drawn from an alphabet of 7 symbols. (142 notes would constitute a rather short piece of music!) Furthermore, all the functions, with the exception of the first which instantiates the symbol S, are sufficiently general (in fact more general than is required for this sequence) to be useful in representing sequences which have a similar recurrence but based on a different pattern, and the "interval" and "ordering" functions, 0–7 and +, are likely to be useful in other contexts also.

Instantiation functions are clearly like the "rewrite rules" common in grammars, but I use the former term because I intend functions to be possibly more complex (even more so than the example above) than is generally the case in grammars. Furthermore, I wish to stress that these functions instantiate abstractions. In the representation above, there are three categories of abstraction: the sequence S, the "intervals" 0–7, and the ordering +. (Note that abstractions are meaningless without their associated instantiation function(s).) If we are designing a computer system to manipulate sequences of the type represented above, we should build the system to operate with these abstractions. This can be justified on pragmatic grounds — our representations of sequences will consume less storage, and processing, which is likely to involve similar use of recurrent interval patterns, is likely to be simpler — and on the grounds of (cognitive) modelling — whatever mechanism (mind) produced these sequences, it is likely to have been governed by constraints of efficiency which imply that the abstractions which lead to this parsimonious representation are likely to have been operative.

To apply this policy in representing music, we must design a set of instantiation functions such that a representation expressed in the abstractions defined by those functions can be instantiated into one or both of the concrete musical products, sound or notation. However, it is not required, or even desirable, that the representation of some sound sequence or page of notation should be instantiated into a precise reproduction of the original. It is enough that the instantiation functions produce an *acceptable* rendition of the represented music, though the definition of what is acceptable will vary. In some cases, for example, sound output on a cheap MIDI synthesiser will be acceptable, in some cases it will not. For a representation to be acceptable the abstractions which are *significant* for the task in hand must be instantiated in a manner faithful to the original whereas other abstractions need not be. If we are not concerned with page layout, for example, a representation of a score need not instantiate the abstract continuous staves (continuing in one long unbroken line, as it were, throughout the piece from start to finish), which are likely to be part of the representation, into the concrete set of short systems of staves found in the original score.

An Illustration: "Temporal" and "Metrical" Representations of Rhythm

As an illustration of the application of the policy of parsimony I present here a comparison of two possible representations of rhythm. No attempt has been made to incorporate a sophisticated representation of pitch into either of these representation schemes, though the possibility of this is not ruled out. The first, which I call the "temporal" scheme, represents a piece of music as a simple sequence of symbols (numbers, perhaps) separated by spaces, with each group of three symbols representing a single note by its onset time, pitch and duration respectively. Notes are represented in order of increasing onset time. The second representation, called the "metrical" scheme, consists of a sequence of symbols or symbol sequences, separated by spaces, representing metrical units all of equal duration. The onset time of a unit is coincident with the end of the previous unit, or immediate if the unit is the first of the sequence. A unit may be a single note, represented by a single symbol indicating its pitch; or a unit may be a group of simultaneous notes, in which case it is represented in the form x,y etc. where x and y are symbols indicating pitch; or a unit may be split into two subunits represented as [X Y], where the sum of the durations of X and Y is the duration of one metrical unit, and the onset time of Y is coincident with the end of X; or a unit may be both split and have one or more simultaneous notes lasting its entire duration, represented for example as [X Y],x,y; or a unit may be null (i.e. consist of no symbol) in which case it represents a rest of one unit's duration. Subunits have the same syntax and semantics as full metrical units, so there can be subunits of subunits etc. The actual duration of metrical units and subunits is given by a "metre indication" at the head of a sequence. The format for this is similar to the format of metrical units so that a number in the metre indication gives the duration of a note occupying an analogous position in the metrical unit. A triple metre, for example, could be indicated by [2 1],3. Where a subunit's duration is not explicitly indicated, it is taken to be half the appropriate unit's duration. Where a note extends across one or more metrical boundaries, it is necessary to represent the note by more than one symbol, in fact one in each of the metrical units through which it lasts. The first of these symbols will have an underscore character "_" following it to indicate that the note continues beyond the end of the metrical unit (similar in function to a tie in music notation). The last symbol will have an underscore before it to indicate that the note begins before the beginning of the metrical unit, and any intervening symbols will have underscores both before and after. (This "metrical" representation scheme has some similarities to the scheme worked out on a wonderfully rational basis by Bernard Bel (1990).)

Properly, one should include the "size" of system necessary to evaluate instantiation functions in the test of a representation's parsimony. In a real computer representation this is quite possible of course, but here I have instead aimed to keep the evaluation of functions simple, hence presumably requiring only a small evaluator. Functions are applied in the order listed. The representation string is searched from its start for a match with the left-hand side of the function. When a match is found the matching segment in the representation string is replaced by the right-hand side. Lower case Roman letters can match any one symbol, including any preceding or following underscore. Lower case Italic letters can match any number of symbols (including none). Upper case Roman letters can match the whole of a metrical unit (which might be null). Where there is a question mark, the predicate following it must be true for the function to be applied. Functions for rational number arithmetic are assumed to have been already defined with priority greater than the functions listed here. Note that spaces are significant but ends of lines are not, and numbers are taken to be single symbols.

The "metrical" scheme is clearly more complex, and so will require larger instantiation functions than the "temporal". In fact something like the "temporal" scheme is likely to be required as an

```
θm\x\→φρ0\m\x\\;
ρt\[xd ye],f\[A B]c →ρt\f\c\ρt\xd\A\ρt+d\ye\B\ρt\[xd ye],f\ ;
ρt\[xd ye],f\[A B]c →ρt\f\c\ρt\xd\A\ρt+d\ye\B\ρt\[xd ye],f\ ;
ρt\d\[A B]c →ρt\d\c\ρt\d/2\A\ρt+d/2\d/2\B\ρt\d\ ;
ρt\xd\ →ρt+d\xd\;
ρt\x\,→ρt\x\;
ρt\xd\p→t p d ρt\xd\;
ρt\x\\→;
φxt p_ dyu _p_ e →xφt p_ u+e-ty;
φxt p_ d yu _p e →xt p u+e-t φy;
φ→
```
Total length: 312 characters

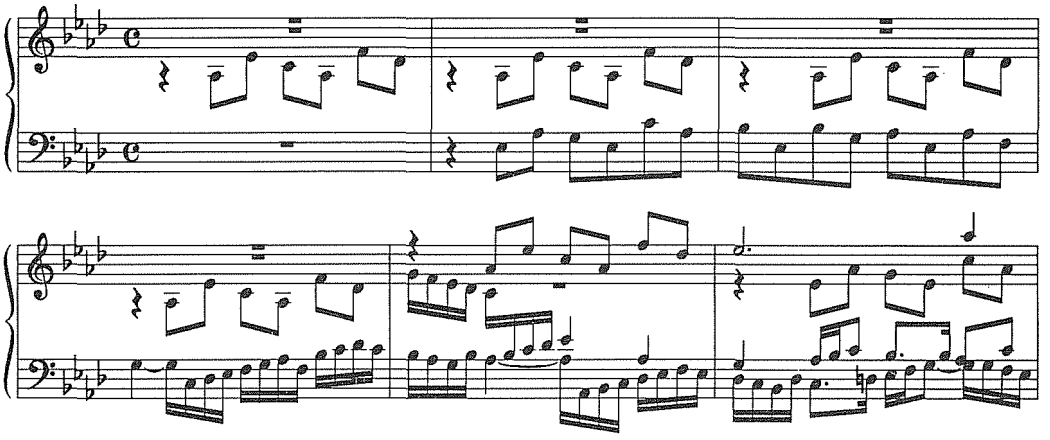
Figure 3. Functions to instantiate a "metrical" representation into a "temporal" one. If x is a "metrical" representation, with a metre indication, θx evaluates to the equivalent "temporal" representation.

intermediate representation on the way to instantiation of the "metrical" scheme in sound, so the functions necessary to instantiate a "metrical" representation into the equivalent "temporal" one are given in Figure 3. On the other hand, the "metrical scheme employs a higher level of abstraction and so is perhaps potentially more parsimonious in the representation of pieces of music. Whether this is so in practice will depend on the nature of the music to be represented.

Figure 4 gives the representation in each scheme of the first two bars of Bach's fugue in A♭ major from Book 1 of *The Well-Tempered Clavier*. Figure 5 similarly gives representations of the first three and a half bars of number VI of Schönberg's *Six Small Piano Pieces,* op. 19.

In all cases the end of a sequence is indicated by a backslash. The "metrical" representations adopt the metrical units suggested by the metre of the notation. (This has been done here for simplicity, and need not be the case in practice. If we are concerned with the representation of these bars alone, for example, the Bach could be notated more parsimoniously adopting a metrical unit equivalent to a crotchet.) The "temporal" representations adopt a time unit of 1=crotchet for the Schönberg and 1=semiquaver for the Bach. We are not concerned with the representation of pitch here, so all pitch representations are counted as if they were a single character to give the lengths indicated in the Figures. In the case of the Bach, then, the second, metrical, representation is clearly more parsimonious. This is not the case for the Schönberg: the lack of alignment of notes with metrical units means that many notes must be represented two or more times, lengthening the second representation to slightly more than the first. (This is not to suggest that the Schönberg is lacking in metrical structure: a representation employing a more sophisticated abstraction of metre might well produce a parsimonious representation of this piece.) If there were not so many simultaneous notes, which can be represented more parsimoniously in the metrical scheme, the difference between the two representations would be even greater.

In useful computer software systems, however, we are concerned not only with representing pieces of music but also operations to be performed on pieces. Here the policy of parsimony operates on two levels. Firstly, an operation can, in principle at least, allow us to represent a piece of music more parsimoniously, as seen above in the illustration of "background" recurrence. Secondly, if we can represent operations performed on the music in question parsimoniously, whether they are implicit in



"Temporal" representation:

```
0 A♭4 2 2 E♭5 2 4 C5 2 6 A♭4 2 8 F5 2 10 D♭5 2 12 E♭5 5 16 E♭4 2 17 D♭5
1 18 C5 1 18 A♭4 2 19 D♭5 1 20 E♭5 1 20 G4 2 21 F5 1 22 G5 1 22 E♭4 2
23 E♭5 1 24 A♭5 1 24 C5 2 25 B♭5 1 26 C6 1 26 A♭4 2 27 B♭5 1 \
```

Total length: 164 characters

"Metrical" representation:

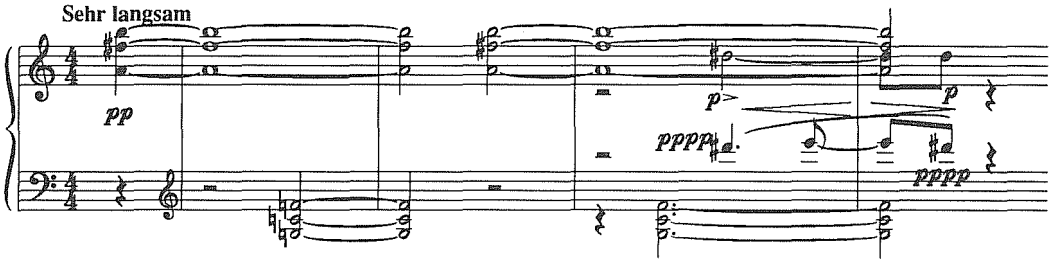
```
16\[[ [A♭4 E♭5]] [[C5 A♭4] [F5 D♭5]]] [[E♭5_ [[_E♭5 D♭5],E♭4 [C5
D♭5],A♭4]] [[[E♭5 F5],G4 [G5 E♭5],E♭4] [[A♭5 B♭5],C5 [C6 B♭5],A♭4]]] \
```

Total length: 93 characters

Figure 4. Representations of the opening of Bach's fugue in A♭ major from book 1 of *The Well-Tempered Clavier*

the composition or explicit from the composer's testimony, relevant theoretical or pedagogical writings, etc., then we have further evidence that our representation's abstractions have cognitive validity.

One simple operation, implicit at the opening of both our example pieces, is to append one segment of music to another. For simplicity's sake, let us consider only operations involving single lines of music for the present (formally, segments of music in which no more than one note is sounding at any one time). This operation is quite easily represented in the temporal scheme. (See Figure 6.) In the metrical scheme the operation can be trivial, since all that is required is to put the representation of one sequence after the other. However, this will not necessarily produce the same result. In the result of the "temporal" operation, the second sequence begins coincidentally with the end of the last note of the first. If one "metrical" sequence is simply placed after the other, the second sequence begins, or more precisely, its first metrical unit begins, coincidentally with the end of the last metrical unit of the first sequence. If the last note of the first sequence does not extend to the end of its metrical unit, or if the first note of the second sequence does not begin at the beginning of a metrical unit (it might, for example, be an anacrusis), there will be a time interval between these two notes which is not reflected in the "temporal" representation. To avoid this time interval, and to represent the type of appending operation which is found at the opening of the Schönberg piece, where the first two chords are



"Temporal" representation:

```
0 B6 7 0 F♯6 7 0 A5 7 3 F5 4 3 C5 4 3 G4 4 7 B6 8 7 F♯6 8 7 A5 8 10
F5 5 10 C5 5 10 G4 5 11 D♯7 1.5 11 D♯6 2.5 12.5 E7 1 \
```

Total length: 103 characters

"Metrical" representation:

```
4\[ [ B6_,F♯6_,A5_]] [ F5_,C5_,G4_],_B6_,_F♯6_,_A5_
[_B6,_F♯6,_A5,_F5,_C5,_G4 B6_,F♯6_,A5_] [[ F5_,C5_,G4_] [D♯7_ [_D♯7
E7_]],D♯6_,_F5_,_C5_,_G4_],_B6_,_F♯6_,_A5_ \
```

Total length: 125 characters

Figure 5. Representations of the opening of no. VI of Schönberg's *Six Small Piano Pieces*, op. 19

followed immediately by a copy of themselves, we must use the much more complex operation shown in Figure 6. This is considerably longer than the equivalent operation in the "temporal" representation scheme.

"Temporal" representation:

```
σxt p d \u→xt p d νt+d-u u;
νn t p d→t+n p d νn;νn \→\
```

Total length: 53 characters

"Metrical" representation:

```
αa_→a_;αa→a_;
ω_a→_a;ωa→_a;
χ[A B] →A B χ;χ\→;χa →αa ωa χ;
σa\ →σa\;
σa[B ] \c\→ψσχa\B \χc\\;
σa\[ B] c\→ψσχa\\B χc\\;
σa\b\→ab\;
ψxa_ _ay ?{_,}⊇{x,y}→xay ψ;
ψA B →[A B] ψ;
ψ\→\;
ψA→[A ]
```

Total length: 176 characters

Figure 6. Functions to append one segment of music to another. If *x* and *y* are representations of segments of music (without metre indications), σ*xy* evaluates to a segment of music in which the first note of *y* is coincident with the end of the last note of *x*

"Temporal" representation:

```
μb\x t p d\b\u?t|b<u→b\x t p u-t|b νt-t|b u;
μb\x t p d\b\u→b\x t p u-t|b+b νt-t|b+b u;
νn t p d→t+n p d νn;νn \→\
```

Total length: 112 characters

"Metrical" representation:

```
μaB \C→aλ\B\C\B\C\;
κa]\→κa\];κa_\→a;
λa\[B _c]\[→λa[\κB\\;
λa\[B C]\[ →λa[B \C\;
λa\b\[→λa[\b\;
λa\b\ c\D\E\→ab c;
λa\B\c\D\E\→D E
```

Total length: 125 characters

Figure 7. Functions to append one segment of music to another. If x and y are representations of segments of music (without metre indications in the case of "metrical" representations), μxy evaluates to a segment of music in which y follows x in its correct metrical position

It has been more frequent in the past four hundred or more years of Western music, however, for composers to have made allowances for metre when appending one segment of music to another. In the Bach piece, for example, the fugue subject (i.e. the opening segment of music up to the first beat of bar 2) appears in many transformations throughout the piece, but always with the same positioning with relation to the metre. (At its final appearance, the subject in fact begins on the fourth beat of the bar rather than the second, but here the pattern of stress has changed, as is clear from the cadences, so that the fourth beat now takes on the role of the second.) Furthermore, the length of the last note varies according to what follows. (Compare, for example, bars 2 and 6.) Thus a more appropriate appending function will firstly preserve the metrical alignment of a segment of music and secondly adjust the length of the final note of the first sequence where necessary. If the final note of the first sequence starts at a metrical position earlier than the first note of the second sequence, the second sequence will start within the final metrical unit of the first sequence, otherwise it will start in the next metrical unit. To construct a function to perform this more sophisticated operation on the "temporal" representation, we must have some means of knowing the metrical position of notes. This is most simply achieved by having a number at the head of the sequence (analogous to the metre indications of "metrical" representations) to indicate how many time units constitute one metrical unit (let us call this the number of "beats"), and to notate onset times so that time 0 is coincident with the beginning of a metrical unit. Times represented by multiples of the number "beats" will therefore also be coincident with the beginnings of metrical units, and we can determine a note's metrical position by taking the remainder of dividing the onset time by the number of "beats" (i.e. the onset time modulo the number of "beats" — in the notation used in the Figures, the modulo operation is indicated by a vertical bar "|"). In the Bach case this would mean putting "16" at the beginning of the representation of the piece, and adding 4 to the onset times shown in Figure 4. The sophisticated appending operation can then be notated for each scheme as in Figure 7. The difference in the lengths of the notations of the operations now all but disappears.

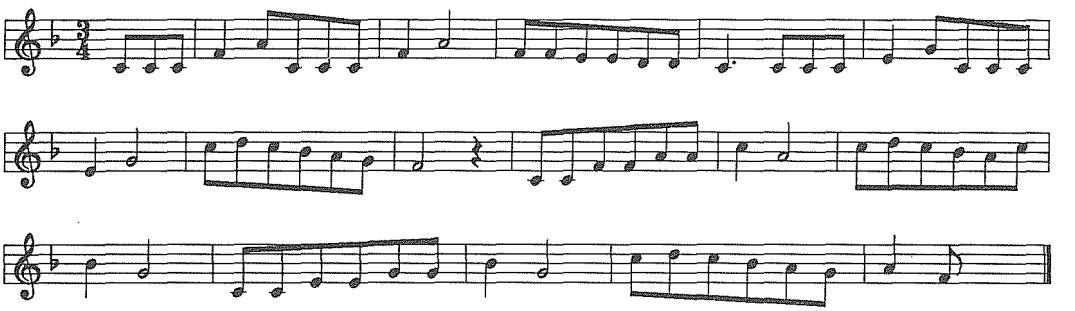


Figure 8. The original version of *La Cucaracha* (from Schaffrath, in press)

A more complex rhythmic operation is implied in two versions of the Mexican song *La Cucaracha.* (See Figures 8 and 9, from Schaffrath, in press.) To return to our first principles, there is clearly a recurrence here, i.e. something in common between the two versions. Furthermore, it is not simply a recurrence of pitches: there is an underlying recurrent pattern in the rhythms also. To express this formally we must define an appropriate function or functions to instantiate this common pattern into the two different versions.

A first approach to this might be to define the operation which converts the original version in triple metre into the better-known version in quadruple meter. Notations for an approximation of this operation are given in each of the representation schemes in Figure 10. (The "metrical" scheme adopts the representation [2 1],3 for triple metre suggested above, i.e. first a splitting into a long plus a short subunit, then a splitting of the long subunit into equal sub-subunits. The notated operations are approximate because the extra note in the eighth complete bar of the better-known version is ignored, and if the corresponding extra note (a crotchet A) is inserted in the original version, the operations do not produce the correct syncopated rhythm for this bar.) The operations are defined so as to operate on polyphonic segments of music, so that if there is any accompaniment to the original version, the resulting version will have a suitably converted accompaniment also. (Thus the function could have some practical purpose: we might have a number of different accompaniments to the original version and want to hear how they sound with the better-known version.)

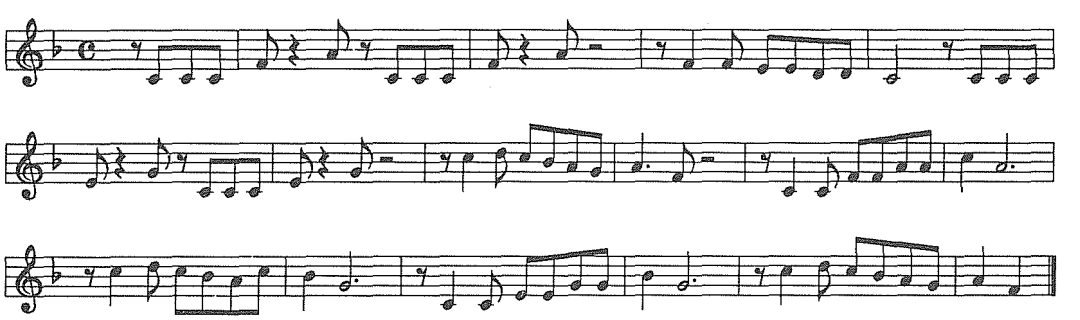


Figure 9. The better-known version of *La Cucaracha* (from Schaffrath, in press)

"Temporal" representation:

```
τb\→b*4/3\ξ7\b/6\0\0\;
ξ7\q\m\n\m p q →n+q p 2*q ξ7\q\m\n\;
n-4*q c q x n-3*q c q y n-2*q c q z ξ3\q\m\n\m p d →n-q p d
ξ3\q\m\n\;
ξ3\q\m\n\m p d ?d≠q→n-2*q p d ξ3\q\m\n\;
ξx\q\m\n\m p d →n p d ξx\q\m\n\;
ξ7\q\m\n\t→ξ2\q\m+q\n+3*q\t;
ξx\q\m\n\t→ξx+1\q\m+q\n+q\t;
ξx\q\m\n\→
```

Total length: 266 characters

"Metrical" representation:

```
τ[2 1],3\→4\ξ;
ξ[[[A B]c D]e F]g →[[ αA\] [ωA\ B]]cαe\ [Dωe\ F]g ξ;
ξ[[A [B C]d]e F]g [[H I]j K]→[Aαe\ [[B C]dωe\ F]]g ξ[[H [ αI\]]j
[[ωαI\ ωI\] K]];
ξA →A ξ;
ξ→;
α[A B]→Aα;α,→,α;αa_→a_α;αa→a_α;α\→;
ω[A B]→Bω;ω,→,ω;ω_a→_aω;ωa→_aω;ω\→
```

Total length: 227 characters

Figure 10. Functions to convert representations of *La Cucaracha*. If *x* is a representation of the original version, τ*x* evaluates to a representation of the better-known version

The "metrical" representation is now a little shorter than the "temporal", mainly because the metrical patterns can be explicitly represented. The "temporal" function has to keep track of the metrical position as it progresses through the sequence, and in the third line has to resort to a long specification of the preceding context in order to produce the syncopation after the repeated-quaver figure. Furthermore, the "temporal" function has some deficiencies compared with the "metrical" function: it does not alter the durations of notes; it is less likely to produce a sensible conversion of any accompaniment; it is not guaranteed to produce a result in which notes are correctly ordered by onset time; it will terminate incorrectly if the accompaniment contains any notes whose onset times do not coincide with quaver beats. To rectify these would require a substantially longer function, making the "metrical" representation clearly the preferred choice for representing music under this type of sophisticated rhythmic operation.

The real advantage of the "metrical" scheme, however, is that it can be used directly to represent the common underlying rhythmic structure of these two versions by employing different metre indications for the two versions, plus a few special cases of the instantiation function ρ to put before those in Figure 3 to produce correct instantiations of some bars of the better-known version which have different metrical structures from the rest. (Note that these will only be invoked on the better-known version because the metre indications of the original version will not match those specified in the special cases.) We need also to introduce "tags" on the initial brackets of the appropriate metrical units to indicate that these special functions are to be used. (The unifier in the evaluator for functions would have to be adapted to ignore these tags where they are not explicitly referred to in the function, just as it

```
θ[2 4],6\Δθ8\Δ:
Δ→[ [[ C5] [C5 C5]]] '[F5 [[A5 C5] [C5 C5]]] '[F5 A5]
[[F5 F5] [[E5 E5] [D5 D5]]] [C5_ [[_C5 C5] [C5 C5]]]
'[E5 [[G5 C5] [C5 C5]]] '[E5 G5] [*[C6 D6] [[C6 B♭5] [A5 G5]]]
'[A5 F5] [*[C5 C5] [[F5 F5] [A5 A5]]] `[C6 A5]
[*[C6 D6] [[C6 B♭5] [A5 C6]]] `[B♭5 G5] [*[C5 C5] [[E5 E5] [G5 G5]]]
`[B♭5 G5] [*[C6 D6] [[C6 B♭5] [A5 G5]]] `[A5 F5] \;
ρt\x8\'A→ρt\[[2 1],3 [[2 1],3 2],5],8\A\ρt\x8\;
ρt\x8\`A→ρt\[2 [4 2],6],8\A\ρt\x8\;
ρt\x4\*[A B]c →ρt\4\c\ρt+1\2\A\ρt+3\1\B\ρt\x4\ ;
```
Total length: 407 characters

Figure 11. Representation of both versions of *La Cucaracha*. When placed before the instantiation functions of Figure 3, the above evaluates to "temporal" representations of the original followed by the better-known versions of the song.

generally ignores underscores.) In contrast to the "conversion" functions of Figure 10, this also allows us to produce the proper rhythm for bar 8 (i.e. syncopated). The two versions of the song can now be represented as in Figure 11 (the A missing from bar 8 of the original version is included), with the common underlying structure represented by Δ. This results in a representation of both versions which, even including the instantiation functions of Figure 3, is 719 characters long. A simple representation (i.e. one without any attempt to represent the derivation of one version from the other) of both versions in the "temporal" scheme, by contrast, would take 924 characters using 1 to indicate the duration of a quaver.

Conclusion

Firstly, let me stress that I am not proposing that either or both of these rather impoverished representation schemes should actually be adopted for use in any particular application — their purpose has been one of illustration only. Secondly, let me formulate from my proposed "policy" an actual procedure for determining the appropriate abstractions to use in designing a musical computer system:

1. Samples of the music to be processed should be gathered. (This is obviously not straightforward nor even possible when the system is intended for use in composition. The design of such systems raises additional issues beyond the scope of this paper.)
2. This repertoire should be scanned for recurrences, whether simple recurrences of musical segments, more complex recurrences of underlying patterns, or even recurrent operations performed on the musical material, whether by the original composer or the potential user of the computer system.
3. Formal representations of these recurrences and of their associated instantiation functions should be designed in some appropriate formal language (e.g. a programming language). No specific procedure can be specified for this step, nor for step 2.
4. The size of representations of both segments of music and instantiation functions should be measured.
5. Other things being equal (there will always be other factors to take into account), those abstractions embodied in instantiation functions which result in small representations should be adopted for use in the design of the system.

References

B. Bel, "Time and Musical Structures", **Interface**, Vol.19, Nos.2–3, 1990, pp.107–135.

E.M. Burns and W.D. Ward, "Intervals, Scales, and Tuning", in D.Deutsch (Ed.), **The Psychology of Music**, Academic Press, Orlando, 1982, pp.241–269.

D.Deutsch, "The Processing of Pitch Combinations", in D.Deutsch (Ed.), **The Psychology of Music**, Academic Press, Orlando, 1982, pp.271–316.

ISO (International Standards Organisation) Committee Draft International Standard "Information Technology — Standard Music Description Language" (ISO/IEC CD 10743), 1991.

J. Kippen and B. Bel, "The Identification and Modelling of a Percussion 'Language', and the Emergence of Musical Concepts in a Machine-Learning Experimental Set-up", **Computers and the Humanities**, Vol. 23, No. 3, 1989.

C.S. Lee, "The Rhythmic Interpretation of Simple Musical Sequences: Towards a Perceptual Model", in P. Howell, I. Cross and R. West (Eds.), **Musical Structure and Cognition**, Academic Press, London, pp.53–69.

A. Marsden and A. Pople, "Introduction: Music, Computers and Abstraction", in A. Marsden and A. Pople (Eds.), **Computer Representations and Models in Music**, Academic Press, London, in press, pp.1–4.

N. Ruwet, **Langage, Musique, Poésie**, Seuil, Paris, 1972.

H. Schaffrath, "The Retrieval of Monophonic Melodies and their Variants: Concepts and Strategies for Computer-Aided Analysis", in A. Marsden and A. Pople (Eds.), **Computer Representations and Models in Music**, Academic Press, London, in press, pp.95–109.

J.A. Sloboda and D.H.H. Parker, "Immediate Recall of Melodies", in P. Howell, I. Cross and R. West (Eds.), **Musical Structure and Cognition**, Academic Press, London, pp.143–167.

Representing Music Symbolically

Mitch Harris Alan Smaill Geraint Wiggins

Department of Artificial Intelligence
University of Edinburgh, 80 South Bridge
Edinburgh EH1 1HN, Scotland
Email:{M.Harris,A.Smaill,geraint}@uk.ac.edinburgh

Abstract

The automated understanding and generation of music is an area of research that raises problems that are central to the Artificial Intelligence enterprise. Recent work at Edinburgh has aimed to use a symbolic AI approach for this field. We indicate how a more abstract understanding of music representation unifies this approach, and how logical descriptions of hierarchical structures can be incorporated. We relate this work to alternative approaches, describe some projects carried out in this framework, and indicate planned future work.

Keywords: Key; Knowledge Representation; Structured Hierarchy; Abstract Data Type.

1 Introduction

... les calculs n'ont cessé d'accompagner la musique tout au long de son histoire, et dans toutes les civilisations ... ([Boulez 85, p 273])

Musicians have an intuitive awareness of the depth of possibilities of musical structure, and how these structures can be manipulated to create new forms of musical understanding. At first sight, computers would seem to be an ideal tool for exploring this mode of expression, yet there seems to be a barrier of *form* between musical ideas and structures that can be implemented on a computer. Our aim is to give ways to overcome this barrier.

There has been relatively little work in the area of AI and music, yet the problems it poses are central to the AI enterprise. There is a particular problem for the traditional symbolic AI approach, as we do not have the benefit of the sort of well developed syntactic and semantic analyses that have been worked out for natural language, for example. On the other hand, while the complementary sub-symbolic approach is effective, and perhaps necessary, for some musical tasks (see section 2.3), such solutions will not give to the user of an AI system the sort of structured understanding and manipulative ability that is usually wanted.

In what follows, we describe how we have attempted to organise formal systems for music representation for manipulation in an automated system. Our intention is to allow a wide range of possible representations, while still permitting the sharing of software, through the use of *abstract data types*. Thus the power of automation may be made available to composers and analysts, by allowing them to think of and access music in terms familiar to them, and yet which also allow automated manipulation.

In section 2, we outline our notion of representation. In section 3, we indicate how an *abstract* approach helps us to deal uniformly with different sorts of representation. In section 4, the topic of higher-level musical structures and their specification is introduced. After describing in section 5 three more extended pieces of work carried out in the style we advocate, we summarise our conclusions and indicate future research directions.

2 What is there to represent?

Before we describe our representation, let us establish what it is that we wish to represent. There are various possibilities, so we will consider them in turn.

2.1 Score / Performance

Often musical databases hold transcriptions of scores, and are intended to record the information in the score rather than to describe some music more directly. The difference can be fairly small for written music in (*eg*) the classical western tradition, but for many other styles the score (when it exists) relates less closely to music as performed or experienced. Therefore we choose to represent in the first instance music *as performed.* This means, for example, that a score that does not determine the order of some musical events will correspond to several possible performances and so to a collection of representations.

2.2 Physical / Psychological

The proces of forming abstractions nearly always has a psychological component. At the lower levels the psychophysics of perception are important; for example the categorical perception of pitch intervals makes possible the abstraction of pitch into note names. Very similar effects occur in our perception of speech; the phoneme *t* does not correspond to any unique pattern of sound energy as this may vary significantly depending on the preceding and subsequent vowel sounds and the position of the consonant in a word. In general, there is no absolute mapping between any physical event and a musical perception, because of the effects of context, categorical perception and the various kinds of filtering which occur due to our biological and neurological make up. See [Handel 89] for discussion of these and related issues.

Having said this, phonemes are a useful abstraction in linguistics, and similarly pitches and durations are useful abstractions for music to use as a base-line for symbolic representation, although the ambiguity between the physical and the psychological is inherent and sometimes problematic. At higher levels, similar considerations will also apply, and either the psychological or the physical phenomena may be the object of interest.

2.3 Symbolic / Subsymbolic

Having decided on an abstraction boundary for our representation (in terms of events with pitches and durations) we can construct formal systems above this layer. One example of such a system is Western tonal music, though this is too restrictive in general. Below this layer we are, by our definition, *outside* the formal system.

To make a mapping between the physical world and the perception of a musical note we need a *subsymbolic* system. This, in turn may be abstractly represented, although it may be

argued that a discrete symbol system is inadequate for this and analogical representations or connectionist systems will be needed ([Leman 88]).

2.4 Analysis / Generation

Our use of a basic level of representation on the level of note-like events is similar to Jackendoff's notion of a "musical surface" [Jackendoff 87, p 217]. But of course this is only a starting point – we want to be able to talk about groupings of these basic events, according to whatever criteria interest us, and about groupings of these groupings, and so on.

We draw a distinction between the unadorned description of the events that make up a piece of music, and whatever richer description interests the listener or composer. Thus on top of the basic description we introduce a mechanism that allows such higher order structures to be introduced in a way appropriate to the task at hand. For example, the analyst may wish to determine metrical or tonal information that is not explicit in the description of the notes of the piece. The mechanism we propose allows the user to represent such higher-level information, either by asserting it directly, or by indicating in general terms the conditions under which these higher-level properties hold. For example, traditional harmonic analysis can be thought of as a series of rules for determining harmonic progressions for music written in a certain style; as such, the rules could be used to indicate what harmonic progression is suggested by given notes, and this could be determined automatically.

In the terms of Nattiez's description in [Nattiez 75], our basic representation is on the level of the "niveau neutre" – it certainly makes no attempt to represent any compositional or affective features that the music may have. However, the mechanism for higher-order structures that we describe in section 4.1 allows the composer to manipulate musical material using structuring devices of the composer's choice, just as it allows a variety of interpretations of the piece to be expressed.

2.5 Representation / Implementation

In what follows we will propose some data structures which we believe are generally useful for reasoning about pieces of music, and which should allow sharing of algorithms for implementations that respect our representation principles. Our proposal is thus *not* of a new software system; indeed, we have implemented parts of this work in several different versions, and in two languages. We envisage that different users will want to describe even notions as basic as pitch and duration in different ways; our proposal is designed to allow this.

3 Abstract Representation

Anyone who looks at computer systems for music representation is quickly struck by the variety of systems on offer. There are some attempts to standardise; for example relatively low-level information may be encoded using MIDI, DARMS *etc* communications protocols. However the possibilities are enormous, and the diversity often prevents us from using software written for one representation for another representation.

Computer scientists have come across this problem in many areas, and have developed a response to it in the use of *abstract data types*. The main idea is that there are patterns

of operations over data that recur frequently in different *concrete* implementations. If the operations are expressed in terms of these general patterns, then they can be made to apply to whatever particular implementation is used, provided we know how to relate the concrete implementation to the *abstract* patterns.

For example, if we consider an operation such as determining the interval between the pitch of two notes, this will be performed in one way if the pitches are represented in Hertz, in another if they are given in number of semitones above middle C, in another if they are given as traditionally notated on a stave, and there are many other possibilities. If we regard all of these as instances of the same computation at an abstract level, this will be one of the operations of our abstract data type. We define such an abstract data type for the descriptions of collections of notes in general. We call this the *basic representation.* This is to be contrasted with particular instantiations of this basic representation in some *concrete* data type.

Once we move beyond this basic representation, there are even more possible operations the user may wish to perform, for example generating music by repeated transformations of some structure that is never explicitly heard (as in "Traum A", section 5.3). We will use the notion of *constituent* to describe in general terms higher-level musical structures that might be so generated, or appear in some analytic process.

We now describe our proposal more formally. The reader not interested in the technical details may skip to section 5.

3.1 The Basic Representation – Specification

The internal structure of our (simplified) event representation is as follows. We define abstract datatypesto represent Pitch (and Pitch Interval, Time (and Duration), Amplitude (and Relative Amplitude) and Timbre, which we will not discuss here – it will be the subject of future work. The abstract event representation is then the cross product

$$\text{Pitch} \times \text{Time} \times \text{Duration} \times \text{Amplitude} \times \text{Timbre}$$

The abstract data type for Time is formed as follows. (The others, except for Timbre, are the same modulo renaming.) The objects of interest are points in time and intervals between them (durations). We want to be able to compute durations from pairs of times, durations from pairs of durations, and so on. We therefore require the existence of functions add_{xy} and sub_{xy} where x and y are each one of $\{$ t d $\}$, standing for Time and Duration, respectively, defining the types of the arguments. The type of the result is unambiguous. Thus, we have functions

$$\begin{aligned}
\text{add}_{dd}&: \text{Duration} \times \text{Duration} \rightarrow \text{Duration} \\
\text{add}_{td}&: \text{Time} \times \text{Duration} \rightarrow \text{Time} \\
\text{sub}_{tt}&: \text{Time} \times \text{Time} \rightarrow \text{Duration} \\
\text{sub}_{dd}&: \text{Duration} \times \text{Duration} \rightarrow \text{Duration}
\end{aligned}$$

There is an ordering on the type Duration, $\leq$. A distinguished symbol for the zero duration, the operation add_{dd} and an inverse are defined so that they make Duration a linearly ordered commutative group. Formally, this means that duration is a Generalised Interval Structure in the sense of [Lewin 87], with extra properties.

The abstract data types for Pitch and Pitch Interval, and Amplitude and Relative Amplitude, may be derived by appropriate renaming of the properties in the above description.

3.2 The Basic Representation – Implementation

Having defined our abstract data type, we now indicate what we mean by the implementation of the type via the provision of corresponding concrete operations.

Each member of a *concrete* event structure is associated with a unique Identifier, for efficient reference by software routines. Such reference is made via destructor functions on the datatypes. We require that the following destructor functions be defined in any instantiation of our representation.

Get$X$ where X is one of { Pitch Time Duration Amplitude Timbre } are unary functions which return the appropriate component of the event tuple associated with the identifier given as their arguments

PutEvent is a function taking an event tuple as its single argument and returning the identifier associated with it

Note that these functions (and those for constituents, specified in Section 4.1) are normally used in conjunction with a database in which the events and constituents making up an idealised performance are stored. The precise form of such a database is immaterial to applications using this representation because of the availability and transparency of the destructor functions.

3.2.1 Example

An example of the event structure, implemented here as a neutral logical term, might use the instances of the abstract data types shown in Figure 1. Timbre is not considered here – we will avoid the issue with ellipsis (...). We associate each event in a represented score with a unique identifier – in this example of the form eN where N is in Integer Number. Finally, in this instance of the representation we notate the existence of the association between the identifier and the appropriate element of the type of data type tuples by the ϵ relation.

Figure 2 shows the specification in the event structure of the first twelve notes of Webern's Variations for Piano, Op 27, as in Figure 3. Note that we do not intend to suggest that this is a readable or person-friendly notation. The attempt is to supply a *standard* (abstract) notation, which may then be translated to user-friendly form *in a general way.*

4 Hierarchical representation

It is widely agreed that a system to manipulate musical representations must allow higher level structures to be introduced hierarchically [Balaban 88, Buxton & *et al* 78]. On the other hand, it is equally widely agreed that such structures and the groupings or kinds of groupings which they delineate must not be imposed on the user of a representation system. Further, it must be possible to assign different hierarchical structures simultaneously to a given set of musical events, in order, for example, that different possible structural interpretations may be represented where there is ambiguity, or that different kinds of information may be represented and related together but kept logically separate (*eg* information about harmony and orchestration).

Pitch	=	{ a b c d e f g } × { ♮ ♯ ♭ } × Integer Number representing traditional Western notation – note name, accidental, and octave
Pitch Interval	=	Integer Number representing integer numbers of semitones
Time	=	Integer Number × Integer Number representing rational numbers of crotchet/quarter-note beats
Duration	=	Integer Number × Integer Number representing rational numbers of crotchet/quarter-note beats
Amplitude	=	Integer Number × Integer Number representing rational numbers of decibels (dB)
Relative Amplitude	=	Integer Number × Integer Number representing rational ratios of Amplitude

Figure 1: An example event instantiation

4.1 The Constituent Representation – Specification

In our representation, we use *constituents* to delineate groupings of events and other constituents. A constituent, at the abstract level, is a pair of the form

⟨Properties/Definition, Particles⟩

Properties/Definition allows logical specification of the relationships between the Particles of this constituent in terms of membership of certain classes, which may be defined externally by the user; the reason for the two-part name for this component will become clear below

Particles is the set of the events and sub-constituents making up this constituent.

A sub-constituent of a constituent is one of its Particles or a sub-constituent of one of them. The Particles of a constituent are restricted so that no constituent may be a sub-constituent of itself. Thus, the constituent structure of a performance in our representation is a directed acyclic graph.

4.1.1 Logical Classification of Constituents

For constituents to be maximally useful, we require that their properties be defined in such a way as to be easily and efficiently available for testing and/or manipulation. That is, given that the user should able to specify whatever musical groupings s/he is interested in, it must be possible to indicate that these groupings have some structural properties – for example, that they are constituted horizontally or vertically with respect to time.

In order to specify Properties more generally, but still allow efficient inference from them, we propose the following approach. The Properties component of a constituent is a pair,

⟨*spec*, *environment*⟩

```
ϵ( e00, ⟨f,♮, 4⟩, 1/4, 1/2, 10/1, ...)
ϵ( e01, ⟨e,♮, 5⟩, 1/4, 1/2, 10/1, ...)
ϵ( e02, ⟨b,♮, 3⟩, 1/2, 1/4, 10/1, ...)
ϵ( e03, ⟨f,♯,3⟩, 3/4, 1/4, 10/1, ...)
ϵ( e04, ⟨g,♮, 4⟩, 3/4, 1/4, 10/1, ...)
ϵ( e05, ⟨c,♯,5⟩, 1/1, 1/4, 10/1, ...)
ϵ( e06, ⟨a,♮, 2⟩, 3/2, 1/2, 10/1, ...)
ϵ( e07, ⟨b,♭, 3⟩, 3/2, 1/2, 10/1, ...)
ϵ( e08, ⟨e,♭, 4⟩, 7/4, 1/4, 10/1, ...)
ϵ( e09, ⟨c,♮, 4⟩, 4/2, 1/4, 10/1, ...)
ϵ( e10, ⟨d,♯, 5⟩, 4/2, 1/4, 10/1, ...)
ϵ( e11, ⟨g,♯,4⟩, 9/4, 1/4, 10/1, ...)
```

Figure 2: Webern, Op 27, bars 1-3

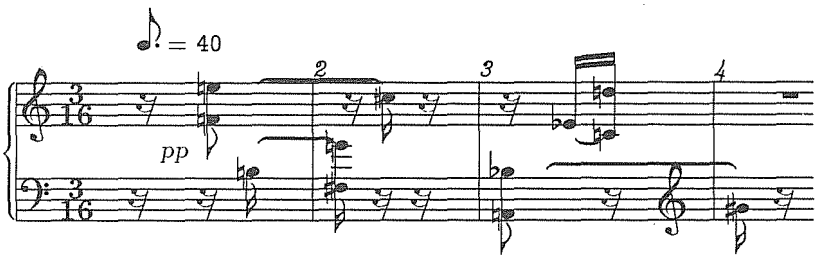


Figure 3: Webern, Op 27, bars 1-4

where *spec* is a logical specification, and *environment* is a (possibly empty) set of values for the result of GetPitch *etc*, when applied to this constituent. The idea is that the *spec* is a logical specification for the defining property of the constituent, which can be checked by looking at the particles of the constituent; the *environment* supplies event-like information on Pitch, Time *etc* associated with the constituent directly.

First, we define the specification language for *specs*. We use a first order logic, with the conventional connectives. We already have the destructor functions on structures and datatypes, and the arithmetic and comparison functions described before (and their derivable relations). Then we can naturally specify the property of (*eg*) a monophonic line, or *stream*, like this, where p_i range over particles:

$$\begin{aligned} &\mathit{stream} \leftrightarrow \\ &\quad \forall p_1.\neg\exists p_2.\ \ p_1 \neq p_2 \wedge \ \ \mathsf{GetTime}(p_1) \leq \mathsf{GetTime}(p_2) \wedge \\ &\qquad\qquad\qquad\qquad\quad \mathsf{GetTime}(p_2) < \mathsf{add}_{\mathsf{td}}(\mathsf{GetTime}(p_1), \mathsf{GetDuration}(p_1)) \end{aligned}$$

This means that the constituent has no particle which starts between the beginning and end of any other particle.

Similarly, we could specify the orthogonal type, the time *slice*, where some point in time is common to every particle in the constituent, like this:

$$\begin{aligned} \mathit{slice} \;\; & \leftrightarrow \\ & \exists t.\forall p_i. \;\; \mathsf{GetTime}(p_i) \leq t \wedge \\ & \qquad\quad t \leq \mathsf{add}_{\mathsf{td}}(\mathsf{GetTime}(p_i), \mathsf{GetDuration}(p_i)) \end{aligned}$$

4.2 The Constituent Representation – Implementation

The implementation of the constituent is very similar to that of the event. The implementation must be a tuple of the form

$$\langle \mathsf{Identifier}, \mathsf{Properties}, \mathsf{Definition}, \mathsf{Particles}, \mathsf{Description} \rangle$$

Identifier is as in the event implementation;

Properties and Definition are, as at the abstract level, a logical specification of the structural properties of the constituent. At the implementation level, though, we make a distinction between properties specified in terms of externally defined predicates and those defined by the constituent itself. More on this below;

Particles are as in the abstraction above;

Description is an arbitrary structure, defined by the user, which is intended for annotation of useful information. Note that no interpretation is given for this component, and that while software may freely write to it, no software *using* its information in any strong sense can guarantee to be portable.

We require that constituents have appropriate typing and destructor functions, as for events.

4.2.1 Properties and Definitions

What, then, is this distinction between the Properties and the Definition of a constituent?

In checking that a constituent has the properties required for meaningful application of a given algorithm, or, indeed, in checking that the Particles of a constituent actually do have the Properties claimed, it will always be necessary to use definitions of any propositions given in terms other than of the basic connectives in the logic and the functions and predicates of the abstract data types. Thus, a definition, external to the constituent structure itself is required – examples for the *stream* and *slice* were given above.

Now, there may well be properties which a user wishes to state about his/her constituent which are simply true of that constituent by definition – for example, that a constituent comprises the events of a particular piece of music. While it is clearly the case that such definitions could be written in the same way as the definitions of *stream* and *slice*, above, it is equally clear that doing so could be arbitrarily laborious – consider, for example, representing a full-scale symphony in this way: the existence of each individual note would need to be verified. This is doubly undesirable by virtue of the fact that all this work has already been done anyway, in writing down the events and constituent structure in the first place.

It makes sense, therefore, at the level of implementation, to distinguish between those propositions which are *derivably* true of a constituent (*ie* its Properties) and those which are *definitionally* true of it. The Definition component contains the latter. As an example, the constituent representing Debussy's "Syrinx", as discussed in [Wiggins *et al* 89, Smaill & Wiggins 90], might be represented thus, where c41 is the identifier of this constituent and e1 ... *e999* are the events comprising it:

(c41, ⟨stream, {}⟩, syrinx, { e1 ...e999 }, "A solo flute piece by Claude Debussy")

In this way, the property *syrinx* is defined to be true of the constituent, but will not be unnecessarily checked by a specification checker, because it is distinguished as a Definition; nor will the user have to give an external definition for it.

4.2.2 Defaults

We specify defaults for each of the Get$X$ functions over constituents as follows, so that we can be sure that, for example, GetPitch will return *some* value which is meaningful, at least to the user who defined the default. The defaults can then be built in to the implementation of the Get$X$ functions. However, it may well in general be undesirable or impossible to specify a universal default system for any given application, and so we allow the specification of over-riding values in the *environment* component of the Properties pair. The *environment* is a set of pairs of named constants and values. The constants are named after the abstract datatypes of the representation (*viz* Pitch, Time, Duration, Amplitude, Timbre), and the associated values are then returned as the result of any call to the corresponding Get$X$ function with this constituent as parameter. For example, consider a constituent, with Identifier c0, whose Properties are defined thus:

$$\langle \text{stream}, \{\text{Time} = 0\} \rangle$$

This constituent has the stream property defined above, and any call of GetTime(c0) will return the value 0.

4.3 Example

A typical example constituent is given in Figure 4 (c00). It records the fact that the events shown in Figure 2 are related together. That they form the subject of the movement is represented by the Definition; and the underlines _ in the Properties and Description positions indicates there are no entries there. In this instance of the representation, we use the relation κ to express that we are dealing with a constituent. The identifier is of the form cN where N is in Integer Number.

Let us now consider how use of the representation given in Figures 1, and 2 allows us to express different readings of the same notes. From the logical descriptions of the constituents it is possible to check or generate constituents with the properties described.

We have already shown how the initial notes may be represented in our notation. In fact this twelve note constituent has the property of containing each degree of the chromatic scale exactly once - that is, it defines a series. We can define what it is for a constituent to have this property, and make the assertion that the first twelve notes form a series in the following form

(figure 4, (c001)). Note that this has the *same* notes as the previous example, but is distinct as a constituent.

Suppose now that we wish to represent some of the internal relations of these notes. For example, if we simply collect together notes sounded simultaneously, we find the four two-note chords we can write as in figure 4 (c03-c06). Then the alternation of two-note chords and single notes can be expressed by a constituent that gathers together the chord constituents and the remaining single notes (c07). Gathering the notes in groups of three in temporal order gives the four triples (c08-c11) (figure 4).

```
κ( c00, _ , subject, { e00 e01 e02 e03 e04 e05 e06 e07 e08 e09 e10 e11 }, _ )
κ( c01, ⟨series,{}⟩, _ , { e00 e01 e02 e03 e04 e05 e06 e07 e08 e09 e10 e11 }, _ )
κ( c03, ⟨chord,{}⟩, _ , { e00, e01 } , _ )
κ( c04, ⟨chord,{}⟩, _ , { e03, e04 } , _ )
κ( c05, ⟨chord,{}⟩, _ , { e06, e07 } , _ )
κ( c06, ⟨chord,{}⟩, _ , { e09, e11 } , _ )
κ( c07, ⟨alternation,{}⟩, _ , { c03, e02, c04, e05, c05, e08, c06, e11 } , _ )
κ( c08, ⟨triple,{}⟩, _ , { e00, e01, e02 } , _ )
κ( c09, ⟨triple,{}⟩, _ , { e03, e04, e05 } , _ )
κ( c10, ⟨triple,{}⟩, _ , { e06, e07, e08 } , _ )
κ( c11, ⟨triple,{}⟩, _ , { e09, e10, e11 } , _ )
```

Figure 4: Constituents for Webern Op 27

Now, the first three of these triples are related in pitch terms – the pitches can be obtained by transposition modulo the octave. This fact can be expressed in a similar way to our other constituents. In this way, the constituent mechanism allows the collection and labelling of events and other constituents.

5 Reasoning with the representation – Some case studies

One approach to Artificial Intelligence in the symbolic tradition aims to simulate "intelligent" behaviour by allowing the goal-directed manipulation of symbolic structures. In presenting our *abstract* basic representation, we have given a *family* of ways of describing music in a neutral manner. In presenting our constituents, we have given ways for users to introduce their own notions of analysis and compositionally interesting features. We now indicate briefly how reasoning takes place via the manipulation of these descriptions so as to permit analysis and generation of music. Reasoning here is thus the inference of higher-level constituent structure during analysis, or of basic structure from generational constituents or from transformations of existing structures.

We now describe some work performed using some of the above ideas. In the first two cases, the use of a particular representation is not central, and our abstract representation framework was used implicitly. In the third case, the representation appears difficult to achieve in other ways, and our representation appears explicitly. [Smaill & Wiggins 90] describes a further example.

5.1 Case study: Small Scale Analysis

In [Wolte 90], Isabel Wolte designed a program for analysis of musical form in small, simple, classical music (minuets and trios by Mozart and Haydn). Based on earlier work by Steedman ([Steedman 77]), a number of procedures were written to deal with the rhythmic, harmonic, and repetition and similarity information all of which is required for the simplest analysis.

The program consists of three relatively independent parts; each part is responsible for one aspect of the analysis and builds up on the information obtained by the preceding part(s). The musical input is first analysed from a rhythmic point of view: a metre for the given piece is suggested and the input reorganised into bars. Then harmonic constituents are evaluated, and regions of specific keys defined, following the ideas of [Longuet-Higgins 62]. A further analysis of these regions establishes the notion of a phrase, punctuated by a cadence, and provides a simple overall structure of the piece. The analysis is performed bottom-up.

In using our representation for such analysis the musical events which we wish to consider are abstractions of a hypothetical listening process; one which anlayses acoustic input and extracts a sequence of pitches and idealised durations (crotchet *etc* or integer numbers of beats) – throwing out (or representing separately) any information about tempo, stress *etc*. A process then acts upon this representation, analysing it to produce constituents representing meaningful units such as cadences. The knowledge of key or time signatures is not assumed and is inferred from the pitch and length information of the musical input.

The program performed well on the restricted range for which it was designed. However, it proved to be brittle outside that range, because of the fixed order of use of the available information. The presence of various experts for different sorts of musical understanding suggests that a blackboard system would work well here, as in [Ebcioglu 88], or a distributed system as suggested by [Minsky 85]. Such systems would share musical representations of the form suggested above.

5.2 Case study: Parsing for Temporal Structure

The ability to determine the internal metrical structure of a piece of music from a performance (or score) with no expressional markers (accents, bar-lines, phrasings *etc*) can be a taught skill – and thus one which seems amenable to formalisation as a parsing process. Where a piece is heard for the first time the parse must happen in temporal order (without lookahead). The inference of higher level metrical structure (time signature and phase) from local events must have a significant 'bottom up' component.

Steedman, in his model of rhythmic analysis [Steedman 73, Wiggins *et al* 89], took the approach of parsing for different metrical feet (*eg* trochees and dactyls) which would only be recognised if they occurred on a strong beat of the metre established up to that point. He had a policy of strict commitment during the parse – the establishment of low level metrical structure always preceded larger (longer) structures, which were never revised once established. This has attractions as a model of listening because it is bottom up and easily implemented as a one-pass process and thus has some psychological plausibility. However, the drawback of sticking to early commitments is that they might turn out to be wrong, leading to misinterpretation of all subsequent structure.

John Whyte has been working on an improved parser which also works in one pass but stores the evidence for different candidate metric structures in a tree, and thus does not suffer from the problem of premature commitment [Whyte 91]. For example, this allows evidence for groupings into two and threes (*hemiola*) to be co-present. There is psychological evidence for such ambiguity ([Handel 89, p 411]). Pitch information can also contribute to this parsing process.

The system acts upon this representation, parsing it to get constituents representing metric chunks. For example, the metrical foot *dactyl* (long-short-short) can be represented as a property thus:

$$
\begin{aligned}
\mathit{dactyl} \leftrightarrow & \\
& \exists p_1.\exists p_2.\exists p_3. \quad \begin{array}{l} \mathsf{add}_{\mathsf{td}}(\mathsf{GetTime}(p_1), \mathsf{GetDuration}(p_1)) = \mathsf{GetTime}(p_2) \wedge \\ \mathsf{add}_{\mathsf{td}}(\mathsf{GetTime}(p_2), \mathsf{GetDuration}(p_2)) = \mathsf{GetTime}(p_3) \wedge \\ \mathsf{GetDuration}(p_2) = \mathsf{GetDuration}(p_3) \wedge \\ \mathsf{GetDuration}(p_1) > \mathsf{GetDuration}(p_2) \wedge \\ \forall p_4.(p_1 = p_4 \vee p_2 = p_4 \vee p_3 = p_4) \end{array}
\end{aligned}
$$

This simply expresses the local constraints within the dactyl itself (the relative durations and order of its particles). In practice higher level constituents are needed to constrain the context in which a dactyl is perceived as such, for example, the note which follows a dactyl must be longer than last two notes of the dactyl itself. However, once satisfactory definitions have been created for the necessary "perceived" constituents, we can experiment with different parsing techniques, and the parser will work for any concrete representation. This illustrates how our representation may be applied to the cognitive modelling of musical processes, using constituents to represent perceived structures.

5.3 Case Study: "Traum A"

"Traum A", for solo computer, by Geraint A Wiggins is an algorithmically generated piece, in the minimalist tradition, which shows two ways in which our representation system can be useful to the composer. In particular, the piece requires the simultaneous representation of notes in equal and just temperament scales.

The piece starts from a "seed", which is repeatedly transformed according to a simple rule. The seed is a set of 128 sine waves, initially played simultaneously, at equal amplitude, each an even-tempered semitone from the next, and centred around Middle C. There is an underlying theme, which, though never explicitly heard, directs the transformations as time proceeds. The theme can be thought of as playing throughout the piece – at any given time, one note will be current. In particular, as well as various time displacements which are not so interesting for our purposes, each transformation step performs pitch and amplitude transformations to achieve the following effect.

Initially, we hear a block of noise, which is effectively unpitched. As the piece proceeds, the even tempered semitones tend in amplitude towards values determined by whichever note of the underlying theme is notionally current. If the pitch of a particular tone is close to a harmonic of that theme note (in the usual "power-of-two" harmonic series), the tendency is towards the amplitude of that harmonic; otherwise it is towards zero. When a tone's amplitude gets close enough to that of the corresponding harmonic, its pitch is altered to be exactly that of the

harmonic. Thus, it has moved from equal to just temperament (with respect to the current note of the theme). Not only does this produce the obvious effect in the harmony of the piece, but as time progresses, the sound we hear tends towards a single note – at the pitch of the current note of the underlying theme – as the non-harmonic tones fade, and the near-harmonic ones become exactly harmonic. The timbre of that single note is then determined by the particular distribution (in the space of frequencies) of the tones of which it is composed.

Because the underlying theme is expressed in equal temperament, each time we pass from one note to the next, there are significant changes in the relative frequencies of the current and target sounds. This gives rise to a constant shifting of the harmonic spectrum, which provides the interest of the piece.

The creative aspect of the piece, then, lies primarily in the choice of theme and of the values determining "closeness" in pitch and amplitude. Similar variables appear in the transformation of the time displacements of each note.

5.4 Representing "Traum A"

There are two ways in which our representation is able to help us with this piece. The first is rather prosaic, and would presumably be possible in the most basic of representations – the obvious way to represent the set of sine waves produced by each step in the transformation process is as events grouped in constituents. Then the transformation step can be represented as a function from constituents to constituents. The resulting constituents can then be bundled together in one constituent representing the whole piece.

More interestingly, we can define a Pitch data-type which has the ability to represent both equal and just temperament at the same time. We require Pitch and Pitch Interval to be defined as in Figure 5. The intuition is that we represent Pitch as some equal temperament base pitch – defined by the first three components of the tuple: note name, accidental, and octave number, as before – with a harmonic multiple and an octave shift (applied *after* the harmonic multiple) to give us access to the just temperament scale related to each equal tempered note in the type. Pitch Interval is then an integer number of semitones (the first component) and a rational multiple which gives the adjustment between the different scales.

Having made these definitions, and supplied the appropriate destructor functions, we can easily supply the seed, values for the two closeness measures, and the theme (which can also be expressed as a constituent). The implementation of the algorithm itself is then almost trivial.

Pitch = { a b c d e f g } × { ♮ ♯ ♭ } × Integer Number × Integer Number × Integer Number
Pitch Interval = Integer Number × Integer Number × Integer Number

Figure 5: Pitch/Pitch Interval datatype for "Traum A"

The whole has been implemented in Prolog, and interfaced to the CSound music programming language via completely general routines based on the abstract representation.

This ease of representation, we claim, would not be available in more conventional notations. In particular, the attempt to notate this piece in conventional score notation would be doomed to failure, unless the composer resorted to the addition of indication *for each and every note*

whether it were in equal or just temperament. What is more, the compositional process itself is aided by the expression of the just tempered scales with respect to bases in the equal tempered scale. This, again, would be difficult and complicated to represent in more conventional terms.

6 Discussion

It is now appropriate to reconsider how our work fits into the broader context of music representation in general. We have already discussed the problem of 'what to represent' in terms of the various dichotomies of score vs performance, physical vs psychological, symbolic vs subsymbolic, analysis vs generation and representation vs implementation. Clearly, there is no panacea, but it is useful to consider the relative merits of different systems along two dimensions — *expressive completeness* and *structural generality*.

'Expressive completeness' simply refers to the range of raw musical data which can be represented, and 'structural generality' refers to the range of high level structures which can be represented and manipulated. For example, at one extreme we can use a waveform to represent any (particular) performance at the cost of being unable easily to capture abstract musical content; even another performance of the same piece may look very different. For another example, MIDI encodings capture some generality about a performance (at the cost of losing some completeness) but do not extend to the expression of general high level structures. Traditional score-based notation has more expressive completeness than MIDI, giving some tonal and metric information, but is restricted in expressive completeness to traditional western tonal music.

Graphical representations of energy spectrum against time (*eg* Xenakis's UPIC) are more useful to musicians than a raw energy waveform but (as yet) there is no uniform way of interfacing these to more abstract structures, particularly those which are not localised in time (*eg* recapitulation). Similar considerations also apply to symbolic representation schemes which give time a privileged status (*eg* [Diener 88, Balaban 88]). However, within the time domain these systems have more *structural generality* than purely graphical representations.

Grammar-based formalisms ([Roads 85]) are also popular. Our suggested formalism is compatible with such approaches, because a parse tree can be expressed naturally in constituent terms. As we have seen above, different representational semantics are possible in this framework, depending on the interests of the user.

With respect to the dimensions of expressive completeness and structural generality, our framework allows more expressiveness than a traditional score, as the mixing of just and even tempered scales shows; it also allows the construction of musically significant structures over any or all the dimensions of the basic representation.

We have shown how our representation may be used for analysis and for composition in a musically intuitive way. We believe therefore that this framework provides an extensible vehicle for automated manipulation of descriptions of musical structures, and as such could allow musicians to gain access to the power of the computer in terms that make sense to the musician. However, this work will benefit in the future from more extensive worked examples, from usage by other musicians and musicologists and, eventually, from the development of software applications.

References

[Balaban 88] M. Balaban. "A music-workstation based on multiple hierarchical views of music", In C. Lischka and J. Fritsch, editors, **Proceedings of the 14th International Computer Music Conference**, pages 56–65, 1988.

[Boulez 85] P. Boulez. "Quoi? quand? comment?", In T. Machover, editor, **Quoi? Quand? Comment?**, pages 272–285. Christian Bourgois, 1985.

[Buxton & *et al* 78] W. Buxton *et al*, "The use of hierarchy and instance in a data structure for computer music" **Computer Music Journal**, 2:10–20, 1978.

[Diener 88] G. Diener. "Ttrees: an active data structure for computer music", In C. Lischka and J. Fritsch, editors, **Proceedings of the 14th International Computer Music Conference**, pages 184–88, 1988.

[Ebcioglu 88] K. Ebcioglu. "An expert system for harmonizing four-part chorales", **Computer Music Journal**, 12, 1988.

[Handel 89] S. Handel. **Listening**, MIT Press, Cambridge, Mass, 1989.

[Jackendoff 87] R. Jackendoff. **Consciousness and the computational mind**, MIT Press, Cambridge, Mass., 1987.

[Leman 88] M. Leman. "Symbolic and subsymbolic information processing in models of musical communication and cognition", **Interface**, 18:141–60, 1988.

[Lewin 87] D Lewin. **Generalized Musical Intervals and Transformations**, Yale University Press, New Haven and London, 1987.

[Longuet-Higgins 62] H. C. Longuet-Higgins. "Letter to a musical friend", **The musical review**, 23:244–8,271–80, 1962.

[Minsky 85] M. Minsky. "Musique, sens et pensée", In T. Machover, editor, **Quoi? Quand? Comment?**, pages 137–63. Christian Bourgois, 1985.

[Nattiez 75] J.-J. Nattiez. **Fondements d'une sémiologie de la musique**, Union Générale d'Editions, Paris, 1975.

[Roads 85] C. Roads. "Grammars as representations for music", In C. Roads, editor, **Foundations of Computer Music**, pages 443–46. MIT Press, 1985.

[Smaill & Wiggins 90] A. Smaill and G. Wiggins. "Hierarchical music representation for analysis and composition", In **Proceedings of the Second International Conference on Music and Information Technology**, Marseilles, France, 1990.

[Steedman 73] M.J. Steedman. **The Formal Description of Musical Perception**, Unpublished PhD thesis, Edinburgh University, 1973.

[Steedman 77] M.J. Steedman. "The perception of musical rhythm and metre", **Perception**, 6:555–69, 1977.

[Whyte 91] J. Whyte. **The automatic rhythmic analysis of monophonic music**, AI/CS Undergraduate Project Report, Edinburgh University, 1991.

[Wiggins *et al* 89] G. Wiggins, M. Harris, and A. Smaill. "Representing music for analysis and composition", In M. Balaban, K. Ebcioglu, O. Laske, C. Lischka, and L. Sorisio, editors, **Proceedings of the 2nd IJCAI AI/Music Workshop**, pages 63–71, 1989.

[Wolte 90] I. Wolte. **Automatic music analysis**, AI/CS Undergraduate Project Report, Edinburgh University, 1990.

The *Interim DynaPiano*: An Integrated Computer Tool and Instrument for Composers

Stephen Travis Pope
The Nomad Group, *Computer Music Journal*, and
CCRMA: Center for Computer Research in Music and Acoustics
Department of Music, Stanford University
P. O. Box 60632, Palo Alto, California USA
Electronic mail: stp@CCRMA.Stanford.edu

Abstract

The *Interim DynaPiano* (IDP) is a tool and instrument for music composition and performance based on a UNIX workstation computer and object-oriented Smalltalk-80 software components. The IDP hardware consists of a powerful RISC CPU with large RAM and disk memories, a hardware-accelerated color graphics system, and interfaces for real-time sampled sound and MIDI I/O. The Musical Object Development Environment (*MODE*) software applications in IDP support flexible structured music composition, sampled sound recording and processing, and real-time music performance using MIDI or sampled sounds. The motivation for the development of IDP is to build a powerful, flexible, and portable computer-based composer's tool and musical instrument that is affordable by a professional composer (i.e., around the price of a good piano or MIDI studio). The basic configuration of the system is consistent with a whole series of "intelligent composer's assistants" based on a core technology that has been stable for a decade. This paper presents an overview hardware and software components of the current IDP system.

Introduction

The *Interim DynaPiano* (IDP) is an integrated computer hardware/software configuration for music composition, production, and performance based on a Sun Microsystems Inc. SPARCstation-2™ computer and the Musical Object Development Environment (*MODE*) software. The SPARCstation is a powerful RISC- (reduced instruction set computer) based workstation computer running the UNIX™ operating system. The MODE is a large family hierarchy of object-oriented software components for music representation, composition, performance, and sound processing; it is written in the Objectworks\Smalltalk-80™ language and programming system.

This paper presents an overview hardware and software components of the current IDP system. The background section below discusses several of the design issues in IDP in terms of definitions and a set of examples from the literature. The hardware system configuration is presented next, and the rest of the paper is a description of the MODE signal and event representations, software libraries, and application examples.

The name *Interim DynaPiano* was chosen for this system for historical reasons. One of the first publications describing the Smalltalk system under development at the Xerox Palo Alto Research Center (PARC) in the 1970's was *Personal Dynamic Media* (LRG 1976), a report that described the Smalltalk-76 system developed by Alan Kay, Adele Goldberg, Dan Ingalls, Larry Tesler, Ted Kaehler, L. Peter Deutsch Dave Robson, et al. running on a Xerox Alto personal computer. The introduction of the report describes in some detail the hardware and software for a "DynaBook" system—a personal dynamic medium for learning and creating—which was unfortunately not implementable at the time. They include a photograph of a mock-up of a "future DynaBook," something that looks astonishingly like a current "notebook" or "tablet" personal computers. Because the name "Alto" was a Xerox-internal code name, the authors were not allowed to use it, so they presented the system they and their colleagues had developed at PARC as an *Interim DynaBook* machine (the Alto PC), with the "interim" software environment Smalltalk-76. There are several sound and music applications included in the examples they present. In the spirit that each machine and each software system is "interim," I decided to dub the system described here *Interim DynaPiano* because it is the first computer-based musical tool/instrument I've built or used that provides a comfortable programming environment and has adequate real-time signal *and* event processing capabilities to be used both as a stand-alone studio, and as a real-time performance instrument.

The IDP and its direct hardware and software predecessors stem from music systems that developed in the process of my composition. Of the MODE's ancestors, *ARA* was an outgrowth of the Lisp system used for *Bat out of Hell* (1983); *DoubleTalk* was based on the Smalltalk-80-based Petri net editing sys-

tem used for *Requiem Aeternam dona Eis* (1986); and the *HyperScore ToolKit*'s various versions were used (among others) for *Day* (1988). In each of these cases, some amount of effort was spent—after the completion of the composition—to make the tools more general-purpose, often making them less useful for any particular task. The MODE, a re-implementation of the HyperScore ToolKit undertaken in 1990-91, is based on the representations and tools used in the recent realization entitled *Kombination XI* (1990) (Part 1 of *Celebration*, work in progress). The "clean-up" effort was minimized here; the new package is much more useful for a much smaller set of tasks and attitudes about what music representation and composition are. If the MODE and the IDP work well for other composers, it is because of its idiosyncratic approach, rather than its attempted generality.

There are two other trade-offs that determine the configuration and usage of a system like the IDP: ease-of-learning vs. power and flexibility; and real-time vs. power. Much modern software is designed to be extremely easy to learn, to the detriment of its power and flexibility. The Macintosh vs. UNIX debate is a prime example of this—the Macintosh can be seen as a computer for children, and a UNIX machine as one for "consenting adults." IDP and the MODE software in particular, is a large and complex system which is non-trivial to learn to use effectively. The hope is that this is more than made up by the power and flexibility of the system. The system is designed to give the best possible real-time performance, but not to be limited by the amount of processing that is doable in real time. The system should provide guaranteed real-time performance for simple tasks (such as MIDI I/O), and should degrade gracefully when real-time processing cannot be achieved.

Background

The development of interactive workstation-based computer music systems started very shortly after personal computers with enough power and flexibility to support such applications became available. By the mid-1970's, several teams had already demonstrated integrated hardware and software tools for real-time software sound synthesis or synthesizer control (e.g., on Alto computers at Xerox PARC, or on early Lisp Machines at the MIT AI Lab).

Integrated Workstation-based Computer Music Systems

The design and configuration of the IDP system described here is the result of several streams of computer music systems. The core technologies of the hardware and software components of IDP are very similar to those used in a number of other computer music workstation systems, both current and in the literature of the last decade (and to my first effort, the Cadmus 9230/m workstation demonstrated at the 1984 ICMC in Paris). These components are:

- commercial engineering workstation (e.g., 680X0- or RISC-based) computer;
- large RAM and soundfile disk memories;
- multi-tasking operating system (e.g., UNIX);
- real-time sampled sound I/O and/or MIDI I/O drivers and interfaces;
- C-based music and DSP software libraries and language (e.g., CARL/cmusic); and
- a flexible, interactive, graphical software development/delivery environment (e.g., Lisp or Smalltalk).

The importance of using a commercial PC or workstation computer is simply wide availability—IDP should not require an electrical engineer to configure it. A powerful multi-tasking operating system with built-in lower-level signal- and event-handling drivers and support libraries is required so that the higher-level software components can be flexible, portable, and abstract. These high-level and front-end components must be written in an interpreted or rapid-turn-around incrementally-compiled software development and delivery environment. The use of a powerful and abstract central programming language integrated with the user interface "shell" is very important to the "dyna" part of an IDP; the goal is to address the issues of learnability, scalability, abstraction, and flexibility, and provide a system that meats the requirements of *exploratory programming systems* as defined in (Deutsch and Taft 1980) or *interactive programming environments* as defined in (Barstow, Shrobe, and Sandewall 1985). The system should also be designed to support interchangeable ("pluggable") interactive front-ends (e.g., graphical editors, or musical structure description languages) and back-ends (e.g., MIDI output, sampled sound processing commands, sound compiler note-lists, or DSP coprocessor control). The components of such packages have been defined (see also [Layer and Richardson 1991] and [Goldberg and Pope 1989]), as:

- a powerful, abstract programming language with automatic storage management, interpreted and/or incrementally-compiled execution, and a run-time available compiler;
- software libraries including reusable low- and high-level modules;

- a windowed interactive user interface "shell" text and/or menu system;
- a set of development, debugging, and code management tools;
- an interface to "foreign-language" (often C and assembler) function calls; and
- a framework for constructing interactive graphical applications.

The primary languages for such systems have been Lisp and Smalltalk-80 (and to a lesser extent Prolog and Forth), because of several basic concepts. Both languages provide an extremely simple, single-paradigm programming model and consistent syntax that scales well to large expressions (matter of debate). Both can be interpreted or compiled with ease and are often implemented within development environments based on one or more interactive *read-eval-print loop* objects. The history of the various (East coast, West coast and European), Lisp machines demonstrates the scalability of Lisp both up and down, so that everything from high-level applications frameworks to device drivers can be developed in a single language system. The Smalltalk heritage shows the development of the programming language, the basic class libraries, the user interface framework, and the delivery platform across at least four full generations. The current Objectworks\Smalltalk system is a sophisticated development environment that is also portable, fast, commercially supported, and stable. Other commercial (e.g., from Digitalk) or public domain (e.g., GNU) Smalltalk systems are largely source-code compatible with "standard" Smalltalk-80.

Two important language features that are common to both Lisp and Smalltalk are *dynamic typing* and *dynamic polymorphism*. Dynamic typing means that data type information is specific to *values* and not *variables* as in many other languages. In Pascal or C, for example, one declares all variables as typed (e.g., `int i;`), and may not generally assign other kinds of data to a variable after its declaration (e.g., `i = "hello";`). Declaring a variable name in Lisp or Smalltalk says nothing about the types of values that may be assigned to that variable. While this generally implies some additional run-time overhead, dynamic binding in a valuable language asset because of the increase it brings in flexibility, abstraction and reusability.

Polymorphism means being able to use the same function name with different types of arguments to evoke different behaviors. Most languages allow for some polymorphism in the form of *overloading* of their arithmetical operators, meaning that one can say `3+4` or `3.1+4.1` The problem with limited overloading is that one is forced to have many names for the same function applied to different argument types (e.g., function names like `playEvent()`, `playEventList()`, `playSound()`, `playMix()`, etc.). In Lisp and Smalltalk, all functions can be overloaded, so that one can create many types of objects that can be used interchangeably (e.g., many different types of objects can handle the `play` message in their own ways). Using polymorphism also incurs a run-time overhead, but, as with dynamic binding, it can be considered essential for a language on which to base an exploratory programming environment for music and multi-media applications.

IDP-like systems of the 1980's

There has been varying progress in the evolution of each of the hardware and software components listed above between the systems developed in 1982 and 1983 and those used today, such as IDP. I will comment on several of the systems that have influenced the current design before presenting IDP in detail. The research systems that appeared along the coasts of the USA in the mid-to-late 1970's were generally programmed in non-mainstream languages and used non-commercial (i.e., non-available) hardware. The advent of UNIX and the Motorola 68000 microprocessor lead to a plethora of commercial UNIX workstation computers in the first years of the 1980's that went by the moniker "3M machines" —1 MIPS CPU performance, 1 MB RAM memory and 1 MPixel display resolution. Several groups used these to develop *computer music workstations, intelligent composer's assistants* or early IDP-like systems.

If I were allowed to write "Interactive Composition Systems I have Known and Loved," it would have to open with a discussion of the SSSP synthesizer developed by Bill Buxton et al. at the University of Toronto in the late 1970's. SSSP combined a DEC PDP-11 with special user interface and sound synthesis hardware (Buxton et al. 1978). The synthesizer could produce up to 16 voices using various synthesis methods in real-time under the control of the PDP-11. The software was a broad range of UNIX-based C routines and applications that all manipulated the same event and voice data structures (Buxton et al. 1979).

The software distribution put together in 1982 by F. Richard Moore and D. Gareth Loy at the Computer Audio Research Laboratory (CARL) at the University of California in San Diego (UCSD) included comprehensive c-language libraries for event and signal processing of all sorts, and the *cmusic*

"sound compiler," a simple, extensible member of the Music-V family. The integration and use of the tools is via the UNIX c-shell, c-preprocessor, and c compiler. The simplicity, comprehensiveness, and extensibility of the CARL software has made it the basis of several powerful computer music research and production tools on a variety of platforms, including DEC VAX, Sun workstations, Apple Macintoshs, and NeXT machines.

The *Cadmus 9230/m* (for music) workstation configuration developed by Günther Gertheiss, Ursula Linder, and myself at PCS/Cadmus computers in Munich in 1983 and 1984 was based on an asymmetrical multiprocessor with 68010 CPUs for the operating system, the window system, the ethernet driver, the file system, and optional terminal or modem packetizing and multiplexing (Cadmus 1985). The DAC/ADC system was based on a (then new) Sony convertor evaluation card with a parallel QBus interface and buffer card. The system had 2-4 MB RAM and 220 MB of disk memory in the default configuration and cost $35,000. Two of these systems were sold, and one of them is still in operation.

Three software environments were implemented on the system; the first was based on my earlier PDP-11-based *mshell* (Pope 1982), an interactive event and signal processing language based on the c-shell and simple windowing, graphics and menu interaction functions. Mshell was a front-end for creating and editing function and notelist files for the *Music11* non-real-time sound synthesis system (using *cmusic* and much better graphics on the Cadmus machine), and provided many synthesis and DSP functions as basic language operators. The second software platform, *ARA*, was based on FranzLisp, the *orbit* object-oriented language, and a Lisp foreign-function call interface to the functions of the CARL software libraries. Starting in late 1984, a series of Smalltalk-80 music languages and tools were built for the 9230/m, leading to DoubleTalk and the HyperScore ToolKit in 1986.

Christopher Fry's *Flavors Band* is a Symbolics Lisp Machine-based tool kit based on the notion of phrase processors that are manipulated and applied in a menu-oriented and Lisp-listener user interface. It used a pre-MIDI connection to a Fender chroma synthesizer for output and was used for a variety of styles and productions. Flavors Band phrase processors served as the models for several current interaction paradigms, such as Miller Puckette's *Max* and the MODE's event generators and event modifiers.

Kyma is a graphical sound manipulation and composition language developed by Carla Scaletti. It is linked to the *Capybara*, a scalable real-time digital signal processor built by Kurt Hebel. The Smalltalk-80-based software tools that comprise Kyma present a uniquely abstract and concrete composition and realization platform. Kyma is one of the only full IDP systems (as defined above) delivered on a personal computer (i.e., PC or Macintosh); it is also a prime example of multi-language integration with Smalltalk-80 methods generating, downloading, and scheduling micro-code for the Capybara DSP, as well as reading and writing MIDI.

Eric Lindemann, Miller Puckette et al's *IRCAM Musical Workstation* (IMW) uses a Next "cube" with a card designed at IRCAM and manufactured by Ariel Corp. that contains two Intel i860 floating-point signal processors and a DSP56001 for I/O. The higher-level software includes performance (e.g., Max), and programming (e.g., Animal) tools. The signal processing capabilities of the system are impressive, but a reasonably-configured system will have a price tag that is more in the Bösendorfer range than IDP.

There are several non-examples I would like to site to further demonstrate the definition of IDP. The various C/C++/Objective C-based systems such as that of the NeXT computer's Sound and Music Kits, the "standard" CARL environment, or the Composer's Desktop Project do provide sound compilers, vocoders and sound processing tools, but fail to integrate them into a fully-interactive exploratory programming environment (see again [Deutsch and Taft 1980]). The range of Macintosh-based MIDI packages includes flexible programmable systems for music (e.g., MIDI-Lisp or HMSL), but these address the event and event list levels only (often using MIDI note events as the only abstraction), and generally rely on MIDI's crude model of signals.

Many other PC- and workstation-based signal-oriented systems generally fall into the categories of closed applications (e.g., the Studer/Editech *Dyaxis*, hard-disk recorders, *Music-N* compiler tool kits, or samplers), that are not programmable "composer's workbenches." Such applications must be provided in an open and customizable way for an IDP to be "dyna."

Smalltalk-80 Software for Music

In the later 1980's, a number of object-oriented frameworks and tool kits for event and signal processing for music (in addition to those already mentioned), were developed and reported in various languages. The Smalltalk-80 literature includes *MusicCategory* (McCall), *SoundKit* (Lentczner), *T-Trees*

(Diener), *VDSP* (Mellinger, Garnett, and Mont-Reynaud), the *NoteEvent* system (Maloney), *TRAX* (Toenne), *AMUSED* (Böcker and Mahling), and *Javelina* (Hebel) (to name quite a few).

The higher-levels of the IDP software are primarily implemented as a set of packages managed as Smalltalk-80 class hierarchies that have evolved (more-or-less) continuously since 1984. The basic class structures and behaviors (and therefore the representation), were modeled after the (1983) ARA system in Lisp, and has gone under the names *SmallSong*, *DoubleTalk*, *HyperScore ToolKit*, *Topaz*, and now *MODE*. We will describe the MODE in more detail below.

IDP Hardware Configuration

Figure 1 shows the configuration of the current mobile IDP system. The general-purpose and audio/MIDI-specific components and specifications are listed below.

- Sun Microsystems Inc. *SPARCstation-2 IPX* workstation, with:
  - 28.5 MIPS, 4.2 MFLOPS, 24.2 SPECmarks CPU performance;
  - 28 MB RAM, 64 kB cache memory;
  - 210 + 660 MB SCSI disk memory;
  - streamer tape cartridge;
  - color graphics accelerator;
  - 16" Sony color monitor, keyboard, mouse;
  - internal CODEC (telephone-quality) DAC/ADC; and
  - Telebit high-speed modem.
- Ariel Corp. *S56X* SBus card with:
  - Motorola DSP56001DSP coprocessor;
  - 64 kB local RAM memory;
  - SBus DMA controller for interface to SPARCstation;
  - programmable Xilinx coprocessor; and a
  - NeXT-compatible (DSP56000 SSI/SCI) DSP-Port connector.
- Ariel Corp. *ProPort* out-board DAC/ADC with
  - pro-audio-quality 44.1 or 48 kHz stereo I/O; and
  - balanced analog audio I/O connectors.

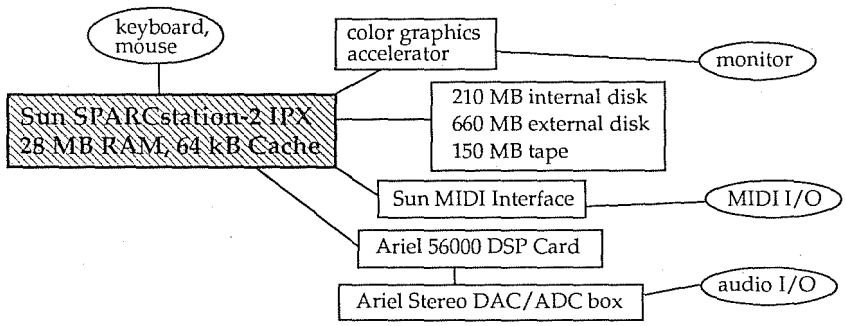


Figure 1: Current IDP hardware configuration

The main differences between this system and the older ones mentioned above is quantitative. The CPU power and memory capacities have increased roughly eight- to ten-fold relative for example to the Cadmus system of 8 years ago, and the dedicated DSP and scheduling facilities have been drastically improved. The hardware system is now small enough to travel (in two flight cases that are approximately 50 cm cubes), powerful enough to execute sophisticated real-time applications, and costs less (around $20,000 at present). The analog audio I/O is also of good enough quality for professional performance or recording. The large RAM (28-40 MB) is chosen in order to support multiple memory-hungry programs (e.g., X/OpenWindows and Smalltalk-80), and still have large in-core sound buffers. This allows near-real-time mixing of multiple soundfiles and simple sampler operation.

MODE: The Musical Object Development Environment

The MODE software system consists of Smalltalk-80 classes that address five areas: (1) the representation of musical parameters, sampled sounds, events and event lists; (2) description of middle-level musical structures; (3) real-time MIDI, sound I/O and DSP scheduling; (4) user interface framework

and components for building signal, event, and structure processing applications; and (5) several built-in end-user applications. We will address each of these areas in the sections below.

Figure 2 is an attempted schematic presentation of the relationships between the class categories and hierarchies that make up the MODE environment. The items in Figure 2 are the MODE packages, each of which consists of a collection or hierarchy of Smalltalk-80 classes. The items that are displayed in `courier` font are written in C and are the interfaces to the external environment. The paragraphs below introduce the packages in the object-oriented style—in terms of the state and behavior of hierarchical trees of related object types.

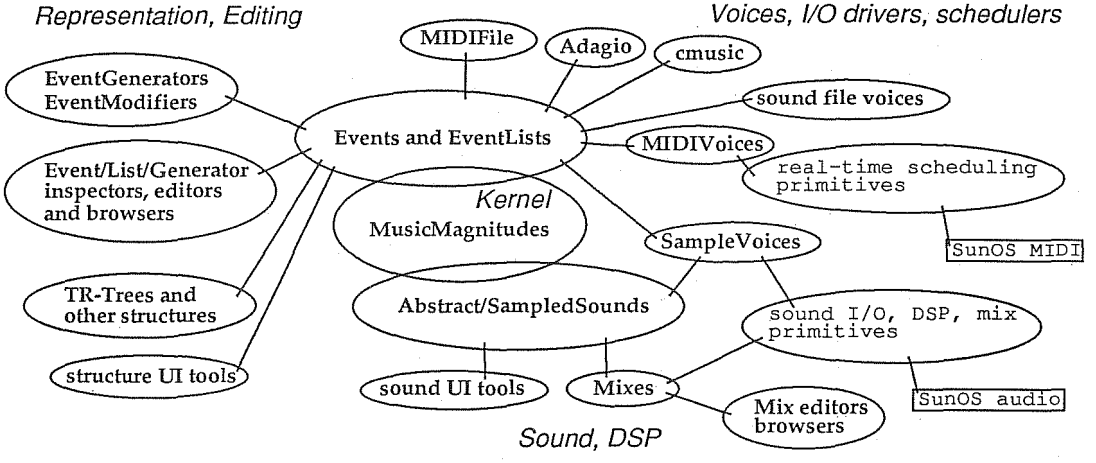


Figure 2: MODE system software components

Using inheritance among Smalltalk-80 classes—the facility to specify software modules (classes) as specialized versions of other modules (their superclasses)—one first defines very general (abstract) classes for the tool kit, then more specific (concrete) subclasses of these that have more specialized messages for their particular behavior. The basic description of the packages, e.g., *Magnitudes*, *Events/EventLists*, *Voices*, or *Sounds*, is the behavior of the abstract classes in each of these categories.

MODE Music Representation

As displayed in Figure 2, the "kernel" of the MODE are the classes related to representing the basic musical magnitudes (such as pitch, loudness, duration, etc.), and for creating and manipulating event and event list objects.

MusicMagnitudes

MusicMagnitude objects are characterized by their identity, class, species, and value. Their behaviors distinguish between class *membership* and *species* in a multiple-inheritance-like scheme that allows the object for "440.0 Hz" to have *pitch-like* and *limited-precision-real-number-like* behaviors. This means that its behavior can depend on *what* it represents, or *how* its value is stored. The mixed-mode music magnitude arithmetic is defined using the technique of *double dispatching* that allows message-passing based on more than one argument (i.e., the receiver of the message). These two methods—membership/species differentiation, and double dispatching—provide capabilities similar to those of systems that use the techniques of multiple inheritance and multiple polymorphism (such as C++ and the Common Lisp Object System), but in a much simpler and scalable manner. All meaningful coercion messages (e.g., `(440.0 Hz) asMIDIKeyNumber`), and mixed-mode operations are defined, e.g., "`440.0 Hz + 7 half-steps`," or "`1/4 Beat + 80 msec`."

The basic model classes include *Pitch*, *Loudness*, *Duration*; exemplary extensions include *Length*, *Sharpness*, *Weight*, and *Breath* for composition or notation specific magnitudes. Figure 3 shows several examples of desirable notations for music magnitudes, along with their value classes and species. The actual class names of these objects (rarely seen by the average user) will be ugly things like *HertzPitch*, *SymbolicLoudness* or *MillisecondDuration*.

The handling of time as a parameter is finessed via the abstraction of duration. All times are durations of events or delays, so that no "real" or "absolute" time object is needed. Duration objects can have simple numerical or symbolic values, as in the examples shown in Figure 3, or they can be conditions

(e.g., the duration until some event *x* occurs), Boolean expressions of other durations, or arbitrary blocks of Smalltalk-80 code. Functions of one or more variables are yet another type of signal-like music magnitude. The MODE *Function* class hierarchy includes line segment, exponential segment, spline segment and Fourier summation functions.

		Magnitude model (species)		
		Pitch	Amplitude	Duration
Implementation model (member)	Integer	36—key number	64—MIDI velocity	500—msec
	Float	440.0—frequency (Hz)	0.7071—ampl. ratio	1.0—sec.
	String	'a5'—note name	'mf'—dynamic name	'1/2'—'beats'
	Fraction	3/2—ratio to 'root'	1/3—ampl. ratio	3/4—'beats'

Figure 3: Examples of MusicMagnitude class membership and species

Events and EventLists

The *Event* object in the MODE is basically just a property-list dictionary with a duration. There are music-specific subclasses that have special knowledge of music magnitudes such as pitch, loudness, and voice. Events have no notion of external time until their durations become active. Event behaviors include duration and property accessing, and "performance," where the semantics of the operation depends on another object—a *voice* or driver as described below. The primary messages that events understand are: `anEvent duration: someDurationObject`—to set the duration time of the event (to some Duration-type music magnitude)—and property accessing messages such as `anEvent color: #blue`—to set the "color" (an arbitrary property) to an arbitrary value (the symbol #blue). The meaning of an event's properties is interpreted by voices and user interface objects; it is obvious that the pitch could be mapped differently by a MIDI output voice and a notation editor. It is common to have events with complex objects as properties such as envelope functions, real-time controller maps, DSP scripts, structural annotation, version history, or compositional algorithms, or with more than one copy of some properties—e.g., one event with enharmonic pitch name, key number, and frequency, each of which may be interpreted different by various voices or structure accessors.

EventList objects hold onto collections of events that are tagged and sorted by their start times (represented as the durations between the start time of the event list and that of the event, thereby avoiding the question of time altogether). The event list classes are subclasses of *Event* themselves. This means that event lists can behave like events and can therefore be arbitrarily-deeply hierarchical, i.e., one event list can contain another as one of its events. The primary messages to which event lists respond (in addition to the behavior they inherit by being events), are `anEventList add: anEvent at: aDuration`—to add an event to the list (sorted by relative start time)—`anEventList play`—to play the event list on its voice (or a default one)—`anEventList edit`—to open a graphical editor in the event list—and Smalltalk-80 collection iteration and enumeration messages such as `anEventList select: [someBlock]`—to select the events that satisfy the given (Boolean) function block. Event lists can map their own properties onto their events in several ways. Properties can be defined as *lazy* or *eager*, meaning whether they map themselves when created (eagerly) or when the event list is performed (lazily). This makes it easy to create several event lists that have copies of the same events and map their own properties onto the events at performance time under interactive control. Voices handle mapping of event list properties via *event modifiers*, as described below.

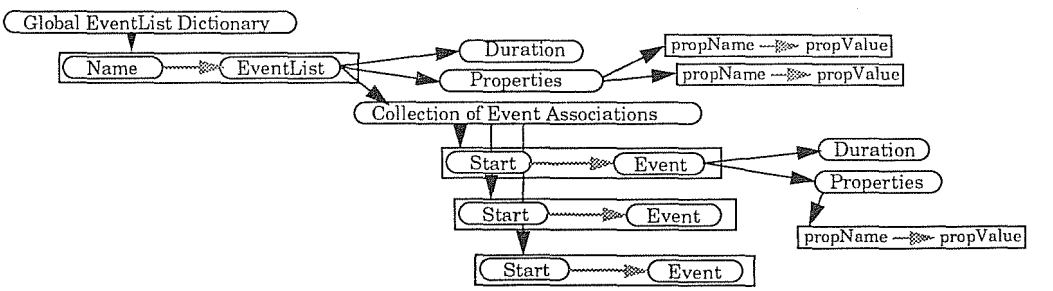


Figure 4: Example Event and EventList state

Figure 4 shows the state of a typical MODE event list as an entity-relationship diagram where the arrows mean *has-a*. One can see the global dictionary for persistent event lists pointing to the given list, which has its own properties (if any), and a sorted collection of associations between durations and

events (or sub-lists). The square boxes and gray arrows represent *associations* between the given name (as in a dictionary key or property name) and the value it "points" to. Each event has its own property dictionary and voice (if desired).

In Figure 5 you see the structure of a typical hierarchical score, where decomposition has been used to manage the large number of events, event generators and event modifiers necessary to describe a full performance. The arrows here mean *is-component-of*. The score is a tree (possibly a forest, i.e., with multiple roots) of hierarchical event lists representing sections, parts, tracks, phrases, chords, or whatever abstractions the user desires to define. The MODE does not define any fixed event list subclasses for these types—they are all various compositions of parallel or sequential event lists.

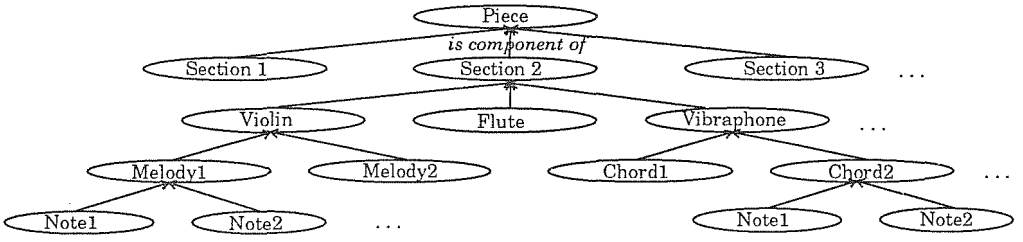


Figure 5: Example score as an event list hierarchy

Note that events do not know their start time—it is always relative to some outer scope. This means that events can be shared among many event lists, the extreme case being an entire composition where one event is shared and mapped by many different event lists (as described in (Scaletti 1989)). There is a small number of event subclasses such as NoteEvent and EventList, rather than a large number of them for different types of input or output media (such as in systems with MIDI, *cmusic*, and DSP event types). The fact that the MODE's simplest textual event and event list description format consists of Smalltalk-80 message expressions (see examples below), means that it can be seen as either a *declarative* or a *procedural* description language. The goal is to provide "something of a cross between a music notation and a programming language" (Dannenberg 1989).

Persistency, Links and HyperMedia

The event and event list classes have two special features that are used heavily in the MODE applications: *persistency* and *links*. Any MODE object can be made persistent by registering it in a shared dictionary under a symbolic name. Browsers for these (possibly hierarchical) dictionaries of (e.g.,) persistent event lists, sounds, compositional structures, functions, or mixes are important MODE tools. MODE objects also have behaviors for managing several special types of links—seen simply as properties where the property name is a symbol such as *usedToBe*, *isTonalAnswerTo*, or *obeysRubato*, and the property value is another MODE object, e.g., an event list. With this facility, one can built multimedia hypermedia navigators for arbitrary network or web types and profiles. The three example link names shown above could be used to implement event lists with version history, to embed analytical information in scores, or to attach real-time performance controllers to event lists, respectively.

EventGenerators and EventModifiers

The EventGenerator and EventModifier packages provide for music description and performance using generic or composition-specific middle-level objects. Event generators are used to represent the common structures of the musical vocabulary such as chords, clusters, progressions, ostinati, or algorithms. Each event generator subclass knows how it is described—e.g., a chord with a root and an inversion, or an ostinato with an event list and repeat rate—and can perform itself once or repeatedly, looking like a Smalltalk-80 control structure. EventModifier objects generally hold onto a function and a property name; they can be told to apply their functions to the named property of an event list lazily (batch or real-time mode) or eagerly (now). The examples at the end of this document present the usage of event generators and modifiers in more detail.

TR-Trees and Other Structures

The last components of the MODE music representation system are the packages the implement higher-level description languages, user interface components, and/or compositional methodologies. The *TR-Trees* system (Pope 1991c) is a rudimentary implementation of some of the ideas in Fred Lerdahl's *Generative Theory of Tonal Music*. The *DoubleTalk* system (Pope 1986) (currently being revived), is a group of interactive tools for editing and executing musical models described as logic-marked Petri nets. Both of these are described in length elsewhere.

MODE Event I/O and Performance

The "performance" of events takes place via *Voice* objects. Events have properties that are independent of the parameters of any synthesis instrument or algorithm. A voice object is a property-to-parameter mapper that knows about one or more output or input formats or external description languages. There are voice "device drivers" for common file storage formats—such as cmusic notelists, the Adagio language, MIDIFile format, phase vocoder scripts—or for use with real-time schedulers and interfacing to MIDI or sampled sound drivers. Some voices also hold onto I/O port objects such as file streams or MIDI drivers; these classes can be refined to add new event and signal file formats or multi-level mapping for MIDI system exclusive messages in an abstract way. Voice objects can also read input streams (e.g., real-time controller data or output from a co-process), and send messages to other voices, schedulers, event modifiers or event generators. This is how one uses the system for real-time control of complex structures.

The *MIDIVoice* classes normally send messages to an object that is a direct interface to the Sun operating system's MIDI scheduler/driver. There is also a Smalltalk-80 version of a simple scheduler for portability, and compatible driver interfaces for other machines. MIDI voices talk to *MIDIPort* objects, so that it is even possible to have multiple MIDI I/O streams, or to use special messages for the specific devices attached to MIDI as mentioned above.

Sampled Sound Processing and Support Classes

The objects that support sampled sound synthesis, recording, processing, and playback are grouped into several packages that support applications in several of the current synthesis/DSP paradigms, including Music-N-style synthesis, graphical interactive sample editing and arithmetic, tape recorders, MacMix-like mixers, and spatial localizers. The basic signal processing language is modeled after *mshell* and presents the model of a pocket calculator with mixed mode arithmetic and graphical inspector/editors on scalar, function and (8-, 16-, 24- or 32-bit) sample array data types. There are interfaces to higher-level analysis/synthesis packages in the form of Smalltalk user primitives (AKA foreign function calls) to C-language routines from Dick Moore and Paul Lansky that implement both phase and linear prediction-based vocoders, as well as sound I/O interfaces to both the 8- and 16-bit audio worlds. The standard sound file formats (e.g., SPARCstation audio, NeXT, IRCAM) are supported for reading and writing different sample formats.

MODE User Interface Components

One of the most powerful aspects of the Objectworks\Smalltalk-80 programming system is the interactive user interface framework, called MVC for *Model/View/Controller* programming (Krasner and Pope 1988). Using MVC means that the user first creates a *model* object—with domain-specific state and behavior (e.g., an event list or a sampled sound). Interactive applications that manipulate instances of this model consist of two objects—the *view* and the *controller*—which can be thought of as output and input channels, respectively. Views generate a textual or graphical presentation of some aspect(s) of the state of the model on the display, and controllers read the input devices and send messages to the model or the view. The advantage of this design is that many applications can use the same set of view and controller components, provided a good library is available.

The Objectworks\Smalltalk-80 development tools and built-in applications reuse a small and well-implemented set of view and controller classes to present the user with various windows that are seen as *inspectors*, *editors*, or *browsers*. An inspector allows one to "see inside" an object and to manipulate its static data interactively by sending it messages. An editor provides a higher-level, possibly graphical, user interface to an object. Browsers organize groups or hierarchies of objects, possibly of the same or similar types, and allow one to select, interact with, and organize these objects into some sort of hierarchy. The MODE user interface is organized as inspectors, editors, or browsers for various families of musical parameters, events, signals, and structures, examples of which will be presented.

Conclusions and Directions

The *Interim DynaPiano* or IDP system is the newest in a series of workstation-based composer's tools and instruments. The system is based on a powerful modern UNIX workstation and a flexible object-oriented software environment. The system costs about $20,000 in its present configuration and is transportable. The bulk of the MODE software described here is available free for non-commercial distribution via anonymous network *ftp* and can be found in the directory `pub/st80` on the InterNet machine named `CCRMA.Stanford.edu`. There are also PostScript and ASCII text files of several other MODE-related documents there, including an extended version of this paper.

The prime advantages of the system are its power and flexibility, and the extensibility and portability of the MODE software. The main problems are the (still relatively high) cost of the system's hardware components, and the complexity of the software. The first of these is a function of time, and the current price-performance war among the engineering workstation vendors can be expected to deliver ever more powerful and cheaper machines in the future.

As of the time of this writing (September, 1991), the system described here is basically functional and in use in realizing the second and third movements of *Celebration*. The components that are still in-progress are the mixing views, the Music-N and localizer classes, and the primitive interfaces to the external vocoder functions (I still use the shell for this). The DoubleTalk re-implementation has yet to be tackled, but should be underway by the time this appears anywhere in print.

The intent of this paper is to present the current IDP system as a member of a family of systems. I believe we can expect the profile of the core technology of IDP systems to remain stable for a number of years to come.

References

I do not cite the full Smalltalk-80, OOP/music or my own reference lists below; real bibliographies for this project can be found in (Pope 1986) and (Pope 1991b); in order to save space, I cite here only the more hard-to-find references and those not cited in the above collections.

Barstow, D., H. Shrobe, and E. Sandewall. 1984. *Interactive Programming Environments.* New York: McGraw Hill.

Böcker, H-D., A. Mahling, and R. Wehinger. 1990. "Beyond MIDI: Knowledge-based Support for Computer-aided Composition." *Proceedings of the 1990 International Computer Music Conference* (ICMC). San Francisco: International Computer Music Association.

Buxton, W. et al.,1978. "An Introduction to the SSSP Digital Synthesizer." *Computer Music Journal.* 2(4): 28-38. Reprinted in Curtis Roads and John Strawn, eds. 1985. *Foundations of Computer Music.* Cambridge: MIT Press.

Buxton, W. et al.,1979. "The Evolution of the SSSP Score Editing Tools." *Computer Music Journal.* 3(4): 14-25. Reprinted in Curtis Roads and John Strawn, eds. 1985. *Foundations of Computer Music.* Cambridge: MIT Press.

Cadmus GmbH. 1985. "Cadmus Computer Music System Product Announcement." *Computer Music Journal.* 9(1): 76-77.

Computer Audio Research Laboratory. 1983. *CARL Software Startup Kit.* San Diego: CARL, Center for Music Experiment, University of California at San Diego, December, 1983.

Dannenberg, R. B., 1989. "The Canon Score Language." *Computer Music Journal* 13(1): 47:56.

Deutsch, L. P., and E. A. Taft. 1980. *Requirements for an Experimental Programming Environment.* Report CSL-80-10. Palo Alto: Xerox PARC.

Goldberg, A. and S. T. Pope. 1989. "Object-Oriented Programming is Not Enough!" *American Programmer: Edward Yourdon's Software Journal.* 2(7): 46-59.

Krasner, G. and S. T. Pope. 1988. "A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80." *Journal of Object-Oriented Programming* 1(3): 26-49.

Layer, D. K., and C. Richardson. 1991. "Lisp Systems in the 1990s." *Communications of the ACM.* 34(9): 48-57.

Learning Research Group (LRG). 1976. *Personal Dynamic Media.* Report SSL-76-1. Palo Alto: Xerox PARC

Lerdahl, F. and R. Jackendoof. 1983. *A Generative Theory of Tonal Music.* Cambridge: MIT Press.

McCall, K. 1980 *The Smalltalk-76 Music Editor.* Xerox PARC internal document.

Pope, S. T. 1982. "An Introduction to *msh*: The Music Shell." *Proceedings of the 1982 International Computer Music Conference* (ICMC). San Francisco: International Computer Music Association.

Pope, S. T. 1986. "The Development of an Intelligent Composer's Assistant: Interactive Graphics Tools and Knowledge Representation for Composers." *Proceedings of the 1986 International Computer Music Conference* (ICMC). San Francisco: International Computer Music Association.

Pope, S. T. 1991a. "Introduction to the MODE." in (Pope 1991b)

Pope, S. T., ed. 1991b. *The Well-Tempered Object: Musical Applications of Object-Oriented Software Technology.* Cambridge: MIT Press.

Pope, S. T. 1991c. "A Tool for Manipulating Expressive and Structural Hierarchies in Music (or, 'TR-Trees in the MODE: A Tree Editor based Loosely on Fred's Theory')." *Proceedings of the 1991 International Computer Music Conference* (ICMC). San Francisco: International Computer Music Association.

Pope, S. T., N. Harter, and K. Pier. 1989. *A Navigator for Unix.* Video presented at the 1989 ACM SIGCHI Conference. Available from the Association for Computing Machinery (ACM), New York.

Toenne, A. 1988. *TRAX Software Documentation.* Dortmund, Germany: Georg Heeg Smalltalk Systeme GmbH.

On Music Making

- Extended Abstract -

Christoph Lischka, GMD St. Augustin (FRG)

One of the most fascinating phenomena in human culture is the emergence of new types of regular (social) behavior, such as the birth of a new scientific paradigm, a new fashion, a new painting style etc.
Hence it is not suprising that a lot of different attempts (philosophy of science, art theory, etc.) have been made to understand how cultural innovation "really" works. In particular, musicology has a long tradition of more or less elaborated endeavors to grasp the "mystics" of musical creation.
Recently, considerable efforts have been made within so-called *Cognitive Musicology* to build up a theory of musical composition[1]; and it seems that a lot of people believe in this type of approach, at least in principle.
In my talk I will argue that these attempts to understand "creativity" are essentially misguided, for deep philosophical reasons; the basic line of research has to be replaced by an alternative ontology and methodology in order to meet the specific demands of the phenomenon in question.

The overall argument is as follows:

- Theorizing about music making always assumes an underlying ontology
- *Artificial Intelligence (AI)* is the outgrowth of a long, rationalistic tradition; its ontology is that of a rational agent, acting in a hostile world.
- This ontology is inadequate as a framework for studying music making: In music, as in other arts, there are musicians, "poetical" human beings, rather than rationalists; their basic concern is not to survive in a hostile world.
- A different, more adequate view is oriented at a poetical picture of Being-in-the-World, which is rooted in the ideas of philosophers like NIETZSCHE, HEIDEGGER, HORKHEIMER, ADORNO, MERLEAU-PONTY, and others. Ontologically, music making is seen as an emergent "social" practice, rather than the outcome of one (or more) individuals.
- Recent developments in *Complex Dynamics* (-> Selforganization) as well as in *Computer Science* (-> Emergent Computation) suggest a methodologically new approach to "Gestaltic" phenomena; they seem to be a natural starting point for a new type of theorizing within such a different ontological framework.
- Two complementary strategies are proposed for theory development. On the one hand, an *empirical analysis* of existing *Music Worlds*, and their dynamics; on the other hand, the development of *computational models* which simulate the complex dynamics of very simple "Music Worlds", e.g. Bird Song.

---

[1] cf. the proceedings of MAI 90, for example

Genealogy of the Rationalist Ontology

Usually, as *Computational Musicologists*, we are concerned either with "structural" properties of musical works, or with their "cognitive processing" by individuals, or with both. Furthermore, as *Cognitive Scientists*, we normally do not reflect upon the underlying assumptions of our discursive practice; we do not ask, for instance, where the subject-object distinction comes from, or similiar questions.

Notwithstanding, from a most abstract point of view, almost all current research in Cognitive Science relies on a rationalist ontology; among its basic features are the following:

- There is a clear distinction between an "external world" of objects, and a "subjective" organism (with a complex interior).
- Between the world and the organism exists a "cognitive" relationship, by which the latter gets a kind of internal representation of the current state of affairs.
- Thinking is a kind of rule-governed internal processing; the rules reside completely within the organism.
- There is a strong decoupling of this highly controlled internal from the (noisy) external dynamics; physical fluctuations are completely suppressed.
- All inter-subjective behavior is (linearly) composed of the atomic behaviors of the individuals (methodological individualism).

As NIETZSCHE and others have argued, this picture of our Being-in-the-World is due to our will to power: Because we felt dominated by nature, we developed a strategy to get its master - we became rationalists.

Towards a New Ontology

In sharp contrast to the common scientific view, many philosophers treat the rationalist attitude as a "deficient mode" of Being-in-the-Word. There exist other ways of being, more basic ones, as, for example, the poet. *Poiesis*, with the genuine meaning of "making", "pro-ducing", is seen by HEIDEGGER as a symmetrical process of "letting things exhibit themselves", where the rationalist separation of passive "stuff" and active "creator" (the subject-object distinction) makes no sense.

I will argue in my talk that, if we are interested in an understanding of *music making*, we have to take up these considerations, and we have to develop a different ontology which makes the underlying intuitions more explicit. This is, admittedly, a long-range undertaking, and it is not restricted to music; but at least we can point to some landmarks:

- As we have already mentioned, there is no clear-cut distinction between "objects" and "subjects". In particular, the dynamics of the world is not the result of actions (of subjects) on objects; rather we have emergent interactional structures[2] which self-organize.
- As a special case, we will treat behavioral regularities as *emergent properties* of ensembles of interacting "things" and "organisms"; we will reject any (metho-

[2]See WALDENFELS, for a more detailed analysis of these concepts

dological) individualism, which usually reconstructs complex behavior as a (linear) composition of atomic units.

- *Music*, as well as any other "cultural" phenomenon, is seen as a kind of "social" practice embedded in a specific world; this excludes from the outset a psychological perspective (which, as a theory of *subjects*, remains within the subject-object distincition).

Self-Organization and Emergent Computation

Historically, several attempts have been made to overcome the rationalist ontology; Gestalt-psychology is probably the most prominent one. But it turned out that holistic, non-rationalist concepts are highly difficult to understand, and that it is nearly impossible to get a broader consensus about their meaning.

Now, it is a suprising fact that *physics itself* has developed holistic concepts recently, well-known as self-organization, synergetics, and others[3]. And there is an increasing community of researchers, applying these ideas to problems in biology, sociology, and psychology; one of the most advanced examples in *Ecological Psychology* is KUGLER/TURVEY.

In the first place, these efforts yield a description of the underlying phenomena which respects much of the intuitions of non-rationalist, "holistic" ontologies. In this sense the approach is non-reductive; on the contrary, it seems that both physics and the non-physical sciences, such as biology, psychology etc., are becoming unified on a level which meets most of the objections of traditional "non-reductionists".

But there remains a difficulty. Because all systems capable of self-organizing behavior are essentially non-linear, a deeper analysis by classical (mathematical) means is impossible. Here so-called *Emergent Computation* gets relevant[4]. By this we mean a new style of experimental computation, where the local dynamics of a complex system is simulated on a fast computing machine and the global behavior can be studied by "simply" looking at the simulation's outcome.

Thus, in addition to a refined conceptual apparatus, we have also a powerful tool for investigating the dynamics of complex systems; both together are a promising starting point to tackle the problems of holism - again.

Approaches to Musicology

Now, how do all these things fit to *Musical Informatics*? It is my opinion that the conceptual framework of self-organization is well-suited to theorize about music making from an ontologically more adequate point of view; furthermore, by the techniques of Emergent Computation we get new methodological tools for exploring complex musical interactions. As a first approximation, then, I would propose two complementary research strategies:

---

[3]See HAKEN and NICOLIS/PRIGOGINE for technical details. EIGEN and HAKEN/STADLER discuss the relationship between the concepts of self-organization and Gestalt.

[4]See HILLIS, for a vividly written exposition of this idea

1. Empirical Studies of Music Making. In contrast to traditional psychological research, but in a line similiar to recent approaches in the theory of the sciences[5], we should try to conceptualize music making as a self-organizing activity: Basically, there are complex organisms, capable of auditive (non-linear) interactions. These organisms are forming groups and societies; from a "macro-perspective", we get global auditive structures (regularities) - musical styles. The individual organisms are constituting this macro-level, but at the same time they are constrained by it also (circular causality). Possible research questions are:
   - How do new musical styles evolve? Or: How does a musical style destabilize, explore the surrounding dynamics, and eventually arrive at a new, stable regime?
   - What is the role of *written* representations of music? How do they change the dynamics of music making?
   - What is the role of *musical instruments*? How do they increase the richness of behavioral structures?

   Of course, these are just a few examples of possible questions; they should only indicate the direction of research. Beyond that, a second line of investigations is suggested:
2. Computer Simulations of Simplified Music Worlds. Probably, we cannot expect to be able to simulate on a level of interesting complexity the dynamics of the development of *human* musical cultures. But, if we carefully reduce the complexity of the phenomena without throwing away the crucial aspects of the problem, we eventually reach systems which are computationally feasible. *Birds* are a highly interesting candidate for investigation: First, there exists a broad empirical background in (neuro-)ethology with respect to the evolution of bird song[6]. Second, by means of modern computational devices, it seems possible to build *colonies of artificial birds*, in order to study their evolution; within such a project, simulations on high-speed computers are an invaluable methodological tool.

References

Clancey, B. 1990. Invited Talk at the DELTA-conference in The Hague, Oct. 1990. Summarized in *AI Communications* 4,1 (March 1991) by J.Sandberg

Eigen, M. 1981. Goethe und das Gestaltproblem in der modernen Biologie. *In*: Rückblick in die Zukunft, ed. by H.Rössner, pp. 209-255. Berlin: Severin und Siedler

Haken, H. 1977. *Synergetics - An Introduction.* Berlin: Springer

Haken, H. and Stadler, M. (eds.) 1990. *Synergetics of Cognition.* Berlin: Springer

Heidegger, M. ^{12}1972 . *Sein und Zeit.* Tübingen: Niemeyer

[5]We point the interested reader to KROHN/KÜPPERS and KNORR-CETINA

[6]See WILLIAMS ET AL., as an introductory survey

Hillis, W.D. 1988. Intelligence as an Emergent Behavior; or, The Songs of Eden. *In*: The Artificial Intelligence Debate, ed. by S.R.Graubard, pp. 175-189

Horkheimer, M. and Adorno, Th. 1947. *Dialektik der Aufklärung*. Amsterdam

Knorr-Cetina, K.D. 1984. *Die Fabrikation von Erkenntnis*. Frankfurt: Suhrkamp

Krohn, W. and Küppers, G. 1989. *Die Selbstorganisation der Wissenschaft*. Frankfurt: Suhrkamp

Kugler, P.N. and Turvey, M.T. 1987. *Information, Natural Law, and the Self-Assembly of Rhythmic Movement*. Hillsdale, NJ and London: Lawrence Erlbaum Ass., Publishers

Merleau-Ponty, M. 1964. *Le visible et l'invisible*. Paris

MAI 90. Proc. of the 2nd "colloque musique & assistance informatique, Marseille 1990" (forthcoming)

Nicolis, G. and Prigogine, I. 1987. *Die Erforschung des Komplexen*. München: Pieper

Waldenfels, B. 1987. *Ordnung im Zwielicht*. Frankfurt: Suhrkamp

Williams, J.M. and Slater, P.J.B. 1990. Simulation Studies of Song Learning in Birds. *In*: Proceedings of the First Conference on Simulation of Adaptive Behavior, pp. 281-287. Paris

Parte 2: Contributi scientifici

Capitolo 1: Reti Neurali

RETI NEURALI PER IL CONTROLLO DELLE DEVIAZIONI TEMPORALI NELL'ESECUZIONE MUSICALE.

R. Bresin, G. De Poli, A. Vidolin

C.S.C. - D.E.I.
Università di Padova
via Gradenigo 6/a, I - 35131 Padova
E.mail ADTPOLI@ipduniv.bitnet
tel. +39 (0)49 8287631 - Fax +39 (0) 49 8287699

ABSTRACT

Musicians' skill has an important role in music performance. What to do with computer music performance? In this paper a connessionist approach to this problem is proposed.
An hybrid system for computer music automatic performance is showed, in which two sets of symbolic and sub-symbolic rules act in cooperation.
Some models of neural nets to calculate time and loudness deviations are discussed.
An application of the system to performe the output of an automatic source of melodies is described.

INTRODUZIONE

I musicisti, a seconda del tipo di strumento, effettuano variazioni nelle intensità, nelle durate e nei timbri delle note, poichè la notazione tradizionale non è sufficientemente precisa riguardo alle intenzioni del compositore. Infatti, se la musica scritta sul pentagramma fosse interpretata seguendo alla lettera le indicazioni metronometriche, risulterebbe estremamente meccanica ed asettica all'orecchio dell'ascoltatore. Sono proprio le deviazioni delle durate delle note rispetto alla loro durata nominale indicata dal metronomo, a rendere un'esecuzione di un interprete musicale diversa da quella di un altro. Oltre alle durate una grande importanza nell'interpretazione è costituita anche dalle variazioni di sonorità, e quindi di pressione sonora: immaginiamo infatti se un pianista, ad esempio, eseguisse una sonata di Beethoven suonando tutte le note con la stessa intensità, l'esecuzione risulterebbe estremamente piatta. I compositori danno delle indicazioni, con una simbologia opportuna, sulle variazioni di sonorità da effettuare da parte dell'interprete, le quali a loro volta non sono seguite alla lettera, così come le indicazioni metronometriche, ma sono "interpretate" secondo il gusto, la cultura e l'abilità dell'esecutore. In sostanza sono le variazioni metronometriche e quelle di sonorità a caratterizzare maggiormente lo stile interpretativo e l'abilità di un esecutore rispetto ad un altro.

In particolare Repp è stato il primo ad analizzare un campione statisticamente rappresentativo di registrazioni commerciali dello stesso brano (Terzo movimento della Sonata per pianoforte op.31 No.3 di Beethoven) eseguito da 19 pianisti di fama mondiale. Da tale analisi si nota che in tutti esecutori le durate dei quarti di ogni battuta hanno subito delle deviazioni, sia in incremento che in decremento, rispetto alle durate metronometriche nominali.

Il problema dell'interpretazione è cruciale nella computer music, dove l'esecuzione è lasciata al computer. Infatti l'esecuzione ottenuta facendo una traduzione letterale delle classiche indicazioni interpretative presenti negli spartiti musicali, porta ad un'esecuzione estremamente meccanica. Si presenta quindi la necessità di formalizzare l'interpretazione così da avere un'esecuzione più "calda", più "umana".

Varie sono state le proposte. Negli anni '30 Carl Seashore e i suoi collaboratori alla Iowa

University (Gabrielsson, 1985) introdussero delle tecniche particolari per lo studio dell'esecuzione pianistica. Venne realizzato un sistema costituito da strisce di balsa incollate ai martelletti. Ognuna delle strisce aveva un foro, che, quando il martelletto veniva colpito, era attraversato da un fascio di luce, che a sua volta impressionava la pellicola di una cinepresa. Con questa tecnica, se pur empirica poichè consentiva un'accuratezza di 0.01 secondi, Seashore fece delle importanti osservazioni: fu notata, ad esempio, una differente durata tra i quarti all'interno di una stessa battuta.

Altri ricercatori hanno cercato di estrarre delle regole interpretative dall'ascolto di esecuzioni di musicisti esperti, come Repp (1990) e Clynes. Quest'ultimo è arrivato ad ipotizzare un andamento metronometrico caratteristico per i maggiori compositori della musica classica occidentale, proponendo delle microstrutture espressive che, introdotte nelle esecuzioni mediante computer, danno all'ascoltatore l'idea sia della personalità del particolare compositore che del suo "stile di movimento" (Repp 1991).

Un'altra proposta è stata quella di De Poli, Irone e Vidolin (1990) che hanno messo a punto un sistema esperto per l'interpretazione di spartiti musicali mediante una base di dati multilivello. In questo approccio sono state utilizzate due basi di conoscenza: una musicale, che codifica l'esperienza e la fantasia dell'interprete, e una sonologica, costituita da un insieme di regole per ogni strumento, che fornisce i parametri fisici dell'esecuzione partendo dallo spartito simbolico.

Un diverso approccio è quello di costruire delle regole simboliche con il metodo di analisi mediante sintesi, proposto da Sundberg e collaboratori (Sundberg et al., 1991; Friberg, 1991). Con tale metodo si sono interrogati musicisti ed insegnanti di musica esperti, che hanno permesso così di formulare delle regole interpretative. Tali regole vengono poi applicate in esecuzione una dopo l'altra secondo l'indicazione di un musicista esperto, il quale giudica se l'effetto aggiuntivo dovuto ad un'ulteriore regola sia gradevole o meno. In pratica l'esecuzione è ottenuta dalla somma delle deviazioni temporali, sonore e timbriche ottenute dall'applicazione contemporanea di un certo numero di regole simboliche.

In questo lavoro si è pensato di utilizzare un sistema ibrido in cui vi siano regole sub-simboliche in eventuale combinazione con regole simboliche. L'idea é di combinare piccole reti controllate da regole simboliche.

REGOLE PER L'ADDESTRAMENTO DELLE RETI NEURALI PER L'ESECUZIONE DI PARTITURE.

Per l'addestramento delle reti neurali é stato utilizzato l'algoritmo di back-propagation, che prevede dei dati in input alla rete e dei dati da imporre al suo output. A questo scopo sono state utilizzate alcune delle regole proposte da Sundberg, considerando i parametri che compaiono in esse come input per la rete, e il risultato, somma di tutte le regole considerate, come output della rete stessa (cioé la deviazione temporale o di loudness, a seconda della rete).

Consideriamo prima la parte sub-simbolica del sistema, in cui si descrive come le reti vengono costruite ed addestrate.

Sundberg e i suoi collaboratori del KTH di Stoccolma (Sundberg et al., 1991) hanno proposto varie regole interpretative ottenute mediante il metodo di analisi mediante sintesi di cui si è parlato in precedenza. Lo scopo delle regole è di convertire la partitura musicale, completa di simboli di frase e di accordi, in un'esecuzione musicalmente accettabile. Le deviazioni risultanti dall'applicazione delle regole sono di tipo additivo, in modo che ogni nota può essere modificata da più regole, e le deviazioni dovuta ad ogni regola sono aggiunte successivamente ai parametri di tale nota. L'ordine con cui tali regole sono applicate non è rilevante, tranne che per le regole di sincronizzazione e quelle che interessano l'ampiezza dell'inviluppo delle note, che devono essere applicate per ultime. Le regole possono essere suddivise in cinque gruppi: regole che interessano un parametro, regole che interessano più parametri, regole di intonazione, regole che agiscono sull'ampiezza dell'inviluppo e regole di sincronismo. Alcune di queste regole sono di segmentazione, per enfatizzare le singole note, altre sono di raggruppamento, che aiutano l'ascoltatore a raggruppare elementi strutturali (temi, frasi, etc.), e tra quest'ultime alcune interessano la partitura ad un livello macroscopico, altre ad un livello microscopico (locale). Infine vi sono delle regole di insieme, quali quelle di sincronismo. La maggior parte delle regole include il parametro k. Esso è usato come peso della regola, per alterarne l'influenza, e dovrebbe essere

usato con lo stesso valore per tutte le regole. Il valore di default è k = 1, ed è appropriato quando tutte le regole sono applicate. Se una regola è applicata isolatamente, allora, secondo Friberg (1991), è meglio utilizzare valori leggermenti più elevati di k per produrre delle variazioni udibili. Inoltre, per ottenere il miglior risultato, per un dato brano musicale, Friberg consiglia di utilizzare un valore di k diverso per ogni regola.

E' possibile quindi utilizzare diversi valori di k per ottenere differenti interpretazioni dello stesso brano musicale, come vedremo più avanti.

Fra tutte le regole proposte da Sundberg e collaboratori abbiamo scelto quelle che interessassero contemporaneamente il minor numero di parametri, per poter lavorare con reti neurali di piccole dimensioni, e che allo stesso tempo fossero tra le più significative dal punto di vista dell'esecuzione musicale. Alla fine si sono considerate 6 regole, che agiscono sulle deviazioni temporali o su quelle di loudness o su entrambi:

1 - contrasto di durata (durational contrast)
2 - doppia durata (double duration)
3 - carico melodico (melodic charge)
4 - intensità elevata (high loud)
5 - durata della nota in un salto (leap tone duration)
6 - salita più veloce (faster uphill)

Le prime quattro regole scelte sono di segmentazione in quanto agiscono isolatamente sulla singola nota, mentre le ultime due sono di raggruppamento.

In particolare le regole 1, 2, 3, 5 e 6 interessano le variazioni di durata della singola nota, e fra esse le regole 1 e 3 insieme alla 4 agiscono anche sulla variazione del loudness, sempre della singola nota.

CODIFICA DEI DATI IN INGRESSO ED USCITA NELLE RETI NEURALI E SCELTA DEI PARAMETRI PER L'APPRENDIMENTO.

Un aspetto fondamentale nell'uso delle reti neurali è il modo in cui si codificano i dati in ingresso e in uscita alla rete. A seconda dei modelli di funzione non lineare utilizzata dalla rete, l'input e l'output devono presentare valori compresi tra 0 e 1 oppure tra -1 e +1. Ad esempio quando la funzione non lineare è la sigmoide, come nel nostro caso, l'input e l'output devono avere valori compresi nell'intervallo [0, 1], mentre se la funzione non lineare è la tangente iperbolica l'intervallo di ammissibilità per i valori è [-1, +1]. In entrambi i casi si considerno compresi gli estremi. Tuttavia, per quanto riguarda l'output, è meglio limitare l'intervallo dei valori ammissibili per non rischiare di mandare in saturazione la rete facendola lavorare in corrispondenza del "ginocchio" della funzione non lineare: ad esempio Rumelhart e McClelland (1988b) consigliano per l'output, nel caso della sigmoide, l'intervallo di valori [0.1, 0.9].

Una volta stabilito il range di valori ammissibili in ingresso ed in uscita dalla rete, è di fondamentale importanza il modo in cui vengono scelti e codificati i dati a disposizione cercando di ricondurli in tale range, ed è proprio da questa codifica e da una sua corretta interpretazione che dipende la riuscita o meno dell'addestramento di una rete neurale.

In primo luogo bisogna assicurarsi che fra tutti i pattern di training utilizzati non ce ne siano di uguali che diano output diversi, perché porterebbero la rete a fornire in output un valore medio anziché quello desiderato. Quindi bisognerà scegliere i dati da dare in input e la loro rispettiva codifica in modo da evitare ogni ambiguità.

Nel caso specifico di questa ricerca, consideriamo ora la prima rete neurale artificiale realizzata, riportata in figura 1. In essa si vede che vi sono quattro neuroni di input e uno di output.

Il primo neurone porta la scritta "Durata Nominale", che sta ad indicare che esso riceve in input la durata della nota così come risulta dalle indicazioni metronometriche della partitura originale. Poiché il neurone accetta in ingresso solo dati tra 0 ed 1 e dal mopmento che le durate utilizzate, per esigenze di calcolo, sono espresse in millisecondi ed essendo la regola del contrasto di durata l'unica, tra le regole scelte, in cui compare la durata nominale della nota in esame e non avendo effetto per note con durata superiore ai 600 ms, si è pensato di dividere per 1000 le durate nominali. Nel caso di note di durata superiore ai 1000 ms, esse vengono forzate ad avere durata 1000 (solamente in ingresso alla rete).

Il secondo neurone di input è quello che tiene conto del carico melodico. In ingresso a tale

neurone viene dato il valore assoluto del carico melodico della nota in esame diviso per il valore assoluto del massimo carico melodico possibile, che è pari a 6.5: $\frac{|Cmel|}{6.5}$, per cui l'input sarà sempre tra 0 (nel caso di carico melodico $C_{mel} = 0$) e 1 (nel caso di $C_{mel} = -6.5$).

Il terzo ed il quarto neurone di input servono entrambi a codificare le informazioni necessarie all'apprendimento della regola di salto. Si sono scelti due neuroni per tale regola dopo varie prove con un solo neurone, in cui ci si è accorti che si incappava in casi di forte ambiguità. Ad esempio salti ascendenti con note di stessa durata e stesso carico melodico davano output diversi. Si è vista allora la necessità di introdurre un ulteriore neurone di input, che tenesse conto del numero di semitoni presenti tra le due note formanti il salto. In particolare, per quanto riguarda il terzo neurone, sono state adottate le seguenti codifiche:

a) input = 0.1: nel caso che la nota in esame sia la prima del salto;
b) input = 0.5: nel caso che la nota in esame non soddisfi alle specifiche della regola del salto;
c) input = 0.7: nel caso che la nota in esame sia la nota di arrivo in un salto discendente;
d) input = 0.9: nel caso che la nota in esame sia la nota di arrivo in un salto ascendente.

Tali valori sono stati ottenuti considerando i coefficienti moltiplicativi presenti nelle regole di salto. Si è osservato che il coefficiente per la prima nota è sempre -4.2 e che quello per la nota di arrivo nel caso di salto ascendente è +4.2, che è il più grande tra i due coefficienti positivi. Allora è stata realizzata una funzione lineare che mappa l'intervallo [-4.2, 4.2] nell'intervallo [0.1, 0.9], ottenendo così che lo 0 dell'intervallo di partenza (che indica l'assenza di salto, poichè come coefficiente moltiplicativo darebbe una deviazione nulla) viene mappato nel valore 0.5 dell'intervallo [0.1, 0.9].

Il valore in ingresso al quarto neurone di input di figura 1 indica il numero di semitoni tra le due note del salto. Tale valore, dovendo essere compreso tra 0 e 1, viene ottenuto dividendo il numero di semitoni per 9 e se il quoziente risulta maggiore di 1, esso viene forzato a valere 1. Cioè si è pensato che nei salti con numero di semitoni superiore a 9, la rete si deve comportare come nel caso di nove semitoni. Infatti, se non si agisse in questo modo, si potrebbero ottenere delle deviazioni temporali esagerate, poichè la radice quadrata del numero di semitoni interviene direttamente come fattore moltiplicativo nella regola del salto (considerando anche che nella moltiplicazione interviene il parametro K, che, come si vedrà in seguito, è stato scelto con valori da 1 a 9).

Anche per la codifica dei valori da insegnare alla rete durante la fase di apprendimento si è utilizzata una funzione lineare del tipo di quella del caso del salto. L'inversione di tale codifica permetterà poi di decodificare l'output della rete dopo la fase di addestramento.

In particolare l'output che viene inizialmente insegnato alla rete, mediante l'algoritmo di back-propagation, è ottenuto applicando le regole simboliche scelte allo spartito in esame (nel nostro caso il Minuetto di Beethoven). Le regole forniscono per ogni nota la deviazione temporale corrispondente ed è proprio questa che viene insegnata alla rete. Pensando di avere al massimo deviazioni di 100 ms, queste vengono divise appunto per 100, ottenendo così valori compresi tra -1 e 1, potendo esserci sia deviazioni positive che negative. Questi valori vengono poi mappati nell'intervallo [0.1, 0.9] dove allo 0 (corrispondente a deviazione nulla) dell'intervallo [-1, 1] corrisponde il valore 0.5 nell'intervallo [0.1, 0.9]: cioè i valori tra -1 e 0 sono stati mappati in [0.1, 0.5] e quelli tra 0 e +1 in [0.5, 0.9].

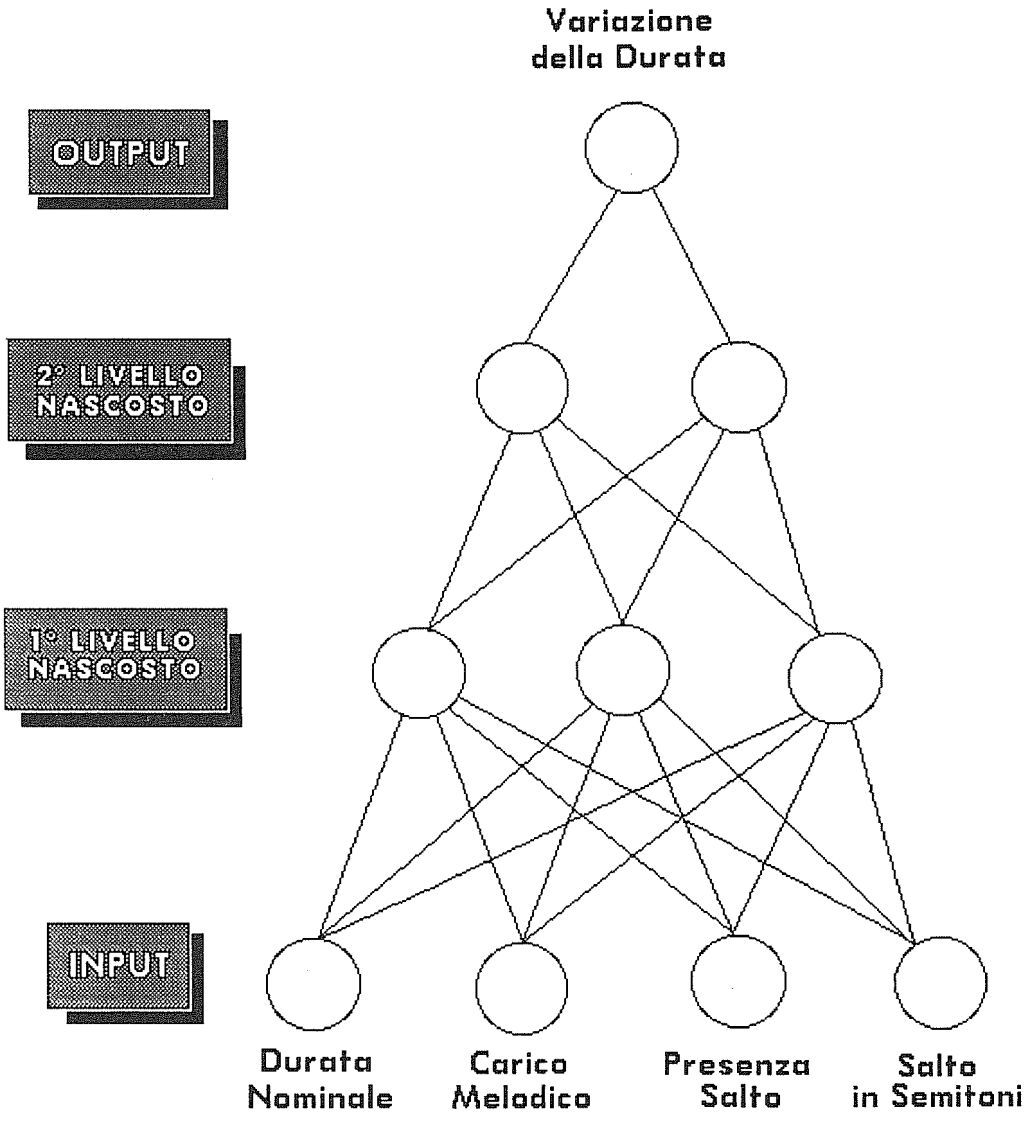


figura 1: Rete neurale per il calcolo delle deviazioni temporali

APPRENDIMENTO DELLE REGOLE

Per insegnare le regole di Sundberg e collaboratori scelte per questo lavoro, è stato considerato il III movimento della sonata per pianoforte op. 31 No. 3 di Beethoven. Si è scelto questo brano in quanto già oggetto di studio da parte di Repp (1990), potendo così confrontare ed insegnare i risultati del presente lavoro con quelli del ricercatore americano.

Il III movimento della sonata in oggetto è così strutturato: Minuetto - Trio - Minuetto. Si veda l'appendice B.

Lo studio in questa tesi è stato focalizzato sulla prima delle due parti di Minuetto. Come esempio per l'addestramento ho preso 16 note, prese sia dal primo ritornello che dal secondo, in modo da fornire il maggior numero di esempi significativi alla rete. In particolare le note scelte sono 16, e corrispondono ai seguenti casi:

- 1 salto ascendente
- 1 salto discendente (di ampiezza diversa da quella del salto ascendente)
- 12 note di durate anche uguali, ma con carichi melodici diversi

Verranno ora discussi alcuni grafici ottenuti testando le reti neurali già addestrate con dei particolari pattern di input (Rumelhart e McClelland 1988b), che mettono in evidenza, come si vedrà in seguito, come la rete abbia effettivamente appreso le regole insegnate (nel caso particolare qui riportato si tratta della regola del salto).

La struttura delle prime reti addestrate è quella riportata in figura 1, in cui vi sono 4 neuroni di input, 1 di output e 2 livelli nascosti. La scelta di tale struttura è motivata dal fatto che con due livelli nascosti, anzichè uno, la rete apprende più velocemente, avendo più gradi di libertà a disposizione per adattare la configurazione dei pesi sulle sinapsi (rami) a seconda del particolare pattern di input. Chiaramente un apprendimento veloce implica anche una opportuna scelta del *momentum* e del *learning rate* , che dopo alcune prove sono stati settati rispettivamente a 0.99 e 0.25.

Per l'addestramento di una rete con tale struttura, utilizzando un insieme di 16 pattern di training come quello descritto in precedenza, sono stati necessari circa 15 minuti di tempo di calcolo di un personal computer con microprocessore 80386 a 33 MHz. Questo è un dato confortante, pensando, come si vedrà e ascolterà dai risultati ottenuti, che uno studente principiante arriva ai risultati interpretativi ottenuti con queste semplici reti almeno dopo un anno di studi.

Risposte della rete al variare del numero dei semitoni tra le note in un salto.

Interessante è vedere come varia l'output quando, con durata = 0 e melodic charge = 0, si impone salto = 0.1, oppure salto = 0.7 o infine salto = 0.9, facendo variare di volta in volta il numero dei semitoni, che viene fatto variare il numero di semitoni tra le due note del salto da 0 a 9 (la scelta di 9 come numero massimo dei semitoni è dovuta al fatto che se si presenta un salto con numero di semitoni maggiore a 9, questo viene forzato a 9 in input alla rete in modo da non avere delle deviazioni temporali esagerate, che invece si avrebbero per valori elevati del numero dei semitoni). Questo è stato fatto per K = 9 e i risultati sono riportati nelle figure 2, 3 e 4. Per salto = 0.1 e salto = 0.9 si vede come l'output abbia andamento rispettivamente decrescente e crescente all'aumentare del numero dei semitoni. Per salto = 0.7 la risposta della rete segue l'andamento crescente previsto dalla regola di Sundberg fino a circa 7 semitoni. Poi la risposta comincia a decrescere, mantenendo comunque una deviazione positiva. Questi andamenti corrispondono in maniera soddisfacente alle relative regole (dove ΔDR indica la variazione di durata e il pedice n indica la prima nota del salto):

nota di partenza: $\Delta DR = -4.2 * \sqrt[2]{\Delta N} * K$ [ms]

nota di arrivo per un salto ascendente: $\Delta DR = 4.2 * \sqrt[2]{\Delta N} * K$ [ms]

nota di arrivo per un salto discendente: $\Delta DR = 2.4 * \sqrt[2]{\Delta N} * K$ [ms]

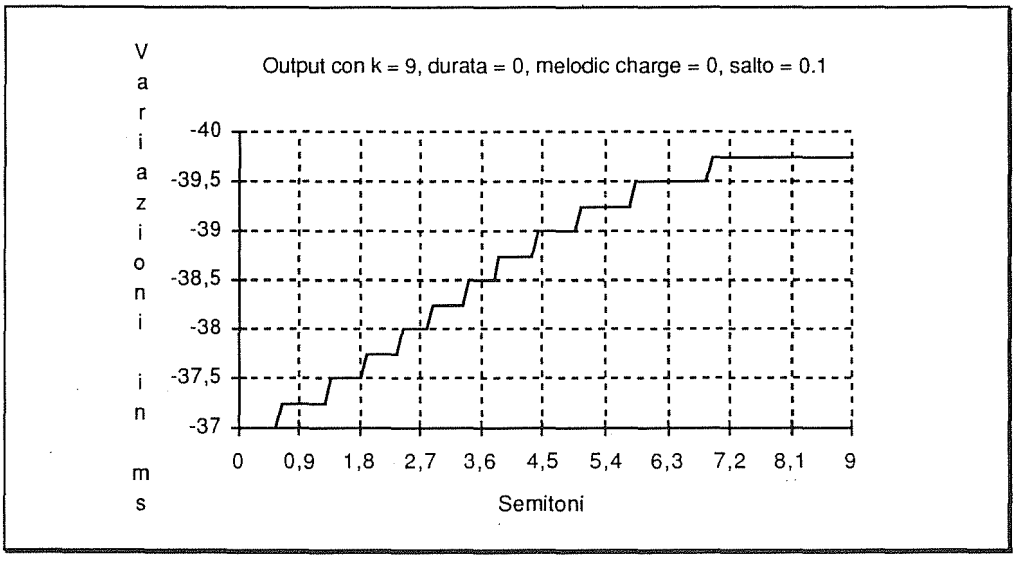


figura 2: Nota di partenza del salto

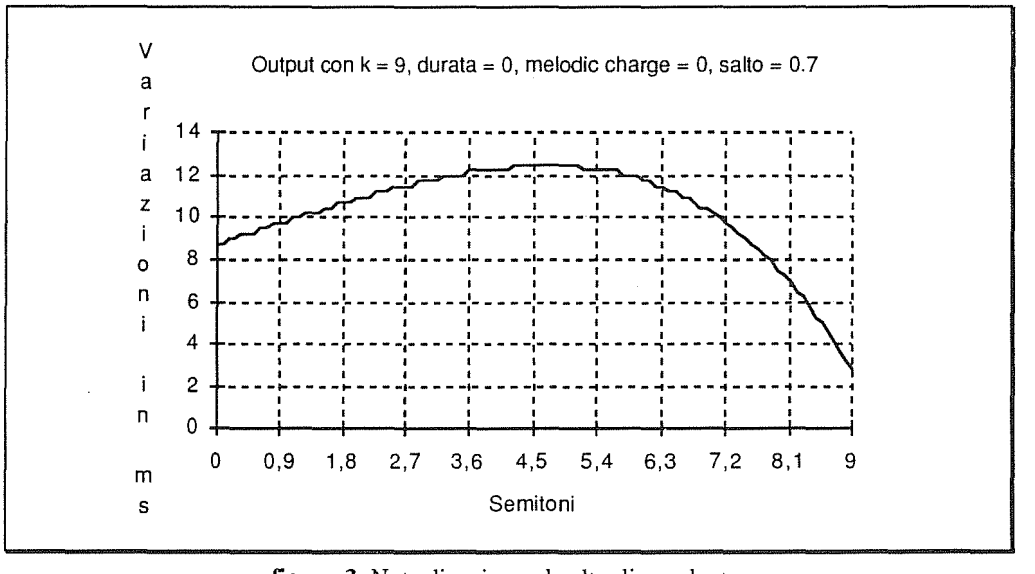


figura 3: Nota di arrivo nel salto discendente

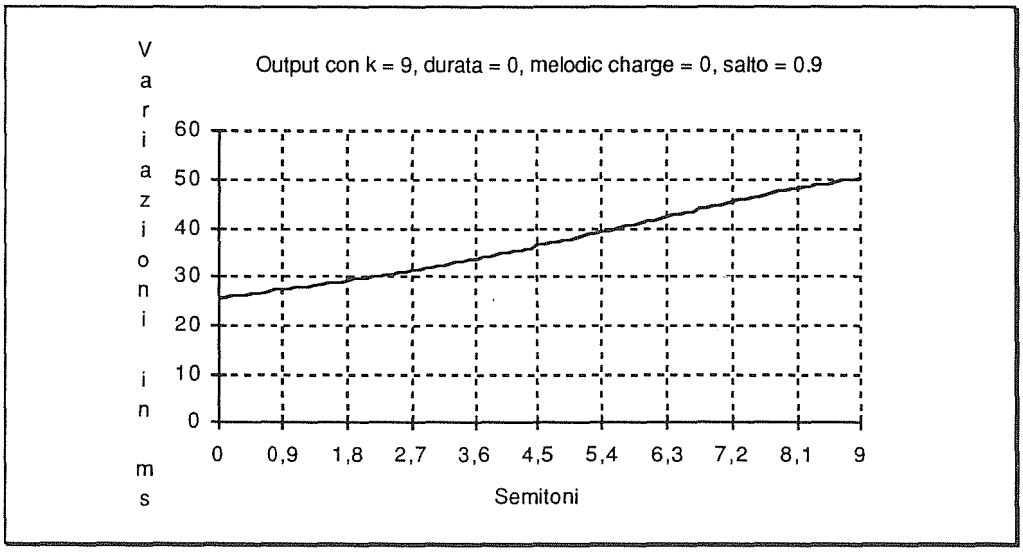


figura 4: Nota di arrivo nel salto ascendente

CONFRONTO TRA LE DEVIAZIONI TEMPORALI OTTENUTE COME OUTPUT DELLA RETE E COME OUTPUT DESIDERATO, NEL CASO DELL'ESECUZIONE DEL MINUETTO.

Dopo l'analisi delle uscite delle tre reti addestrate per K = 1, K = 5 e K = 9 è interessante sottolineare che nell'esecuzione dell'intero minuetto di Beethoven le reti hanno risposto benissimo per note di durate che non erano state apprese durante la fase di learning. Inoltre, soprattutto per la rete addestrata con K = 9 , la deviazione temporale ha mostrato un andamento abbastanza diverso da quello previsto dalle regole di Sundberg in corrispondenza dei salti. In particolare lì dove, in corrispondenza della nota di partenza del salto, le regole di Sundberg forniscono una deviazione temporale negativa piuttosto accentuata, la rete tende a diminuire, in valore assoluto, tale decremento. Dove invece le regole, sempre in corrispondenza della nota di partenza del salto, darebbero un incremento della durata della nota, in seguito alla concomitanza dei contributi con deviazioni positive dovuti alle regole diverse da quella del salto, la rete dà un decremento o comunque un incremento minore di quello ottenibile con le regole simboliche. Inoltre, nel caso di prima nota del salto, ove il decremento sia di piccola entità, questo viene leggermente aumentato in valore assoluto.

Invece, in corrispondenza della nota di arrivo di un salto ascendente, la deviazione fornita dalla rete è uguale o leggermente superiore a quella che si otterrebbe con l'applicazione diretta delle regole di Sundberg. Nel caso del salto discendente, invece, la deviazione temporale fornita dalla rete è sempre leggermente superiore a quella che si otterrebbe dalle regole simboliche di Sundberg.

Quindi, in sostanza, il risultato più interessante è quello che interessa la nota di partenza in un salto tra due note. Infatti dai risultati ottenuti è chiaro che la non linearità intrinseca nella rete neurale operi con un'azione di "smussamento" fornendo dei risultati migliori di quelli ottenibili mediante un'applicazione additiva, spesso esagerata, delle regole simboliche di Sundberg. Inoltre, quando le regole simboliche forniscono un incremento della durata della prima nota del salto, la rete neurale, cioè le regole sub-simboliche, tendono a fornire un decremento o un incremento pressoché nullo, indicando quindi che il "concetto di decremento della durata in corrispondenza alla prima nota di un salto" è stato appreso molto bene dalla rete.

SISTEMA IBRIDO

In analogia con lo schema di figura 6 (Hubel, 1975) si può pensare di rappresentare il sistema utilizzato per il presente lavoro come in figura 5, dove tutto ciò che sta a monte dei livelli nascosti della rete neurale corrisponde ai livelli da A a C di figura 6 e tutto ciò che stà a valle è analogo ai livelli da X a Y della stessa figura.

In pratica potremmo pensare che il livello dei neuroni di input corrisponde ai nervi ottici del pianista stimolati dalla lettura della partitura (causa), mentre il livello di output corrisponde all'effetto, quindi all'azione meccanica, corrispondente al particolare input.

La partitura può essere acquisita mediante un sintetizzatore collegato ad un personal computer tramite interfaccia MIDI, oppure con un editor musicale o mediante un opportuno file dati. A questo punto la partitura viene sia analizzata dalle due regole simboliche utilizzate in questo lavoro che produrranno un particolare output (OUT_2 in figura 5), che codificata in maniera opportuna, così da poter essere "letta" dalla rete neurale. Quest'ultima, in seguito all'applicazione delle regole sub-simboliche, fornirà a sua volta un output che, opportunamente decodificato (OUT_1 in figura 5), verrà sommato a quello fornito dalle regole simboliche. Il risultato della sommatoria verrà poi applicato alla partitura originale per dare la partitura "esecutiva", cioè la partitura ottenuta con l'applicazione delle regole. Quindi la nuova partitura viene inviata, attraverso l'interfaccia MIDI, al sintetizzatore per il suo ascolto. Inoltre, un'analisi mediante regole simboliche della partitura permette di determinare (nel caso del Minuetto in esame) se si sta eseguendo o no il ritornello, facendo agire, a seconda dei casi (freccia tratteggiata), una rete oppure l'altra (addestrate ad esempio con un diverso k, o con il diverso stile che il pianista adotta a seconda dei casi) oppure scegliendo il valore di k (come si è visto nelle reti con 5 neuroni di input).

Analizziamo il sistema nelle sue parti principali.

Regole simboliche.

Le regole simboliche utilizzate in questo sistema sono due ed agiscono entrambe sulla variazione della durata della singola nota. Esse sono le regole 2 e 6 riportate nel capitolo 3. Si è pensato di non insegnare queste regole alla rete neurale perchè estremamente semplici e di applicazione immediata.

Regole sub-simboliche.

Per l'applicazione di queste regole si sono utilizzate in un primo momento due reti neurali: una per ottenere le deviazioni temporali e l'altra per le deviazioni di loudness (trasformate in una variazione di key velocity).

La rete utilizzata per insegnare le regole di deviazione temporale è quella riportata in figura 7. L'addestramento utilizzato è stato quello mediante l'algoritmo di back propagation, in cui al neurone di output è stato imposto, di volta in volta, il valore ottenuto mediante l'applicazione delle tre regole adoperate.

Al primo neurone di input viene assegnato il valore della durata effettiva della nota così come riportato nella partitura originale.

Al secondo neurone di input viene assegnato il valore del carico melodico della nota in esame, così che essa viene inquadrata nel particolare contesto armonico.

Gli ultimi due neuroni di ingresso sono stati utilizzati per insegnare alla rete la regola del salto. In particolare il primo dei due viene utilizzato per indicare se la nota in esame è quella di partenza o di arrivo nel salto, e nel secondo caso si distingue con un'opportuna codifica se si tratta di un salto ascendente o discendente. Il secondo di questi due neuroni indica la lunghezza del salto in semitoni.

Tutti i valori in ingresso sono opportunamente scalati tra 0 e 1, poichè la funzione non lineare adottata dalla rete utilizzata è la sigmoide.

Il nodo di output fornisce la deviazione della durata per la particolare nota in esame. Anche il valore dell'output è scalato, questa volta tra 0.1 e 0.9 per evitare di mandare in saturazione la rete: al valore 0.5 corrisponde una deviazione nulla, per valori compresi tra 0.1 e 0.5 si ha un accorciamento della durata nominale e per valori tra 0.5 e 0.9 un aumento di essa.

Sono state addestrate tre reti neurali come questa, utilizzando per ognuna un diverso valore del

parametro k di pesatura delle regole: in particolare si sono usati i valori k = 1, k = 5 e k = 9. Per k = 1 si ottengono delle variazioni difficili da udire, mentre con k = 9 , essendo maggiormente amplificate le regole, si hanno delle deviazioni temporali molto marcate.

In un secondo momento è stata realizzata un'altra rete neurale con 5 neuroni di input anzichè 4 come in precedenza. I primi 4 nodi ricevono gli stessi input della rete precedente, mentre al 5° viene assegnato il valore del parametro k. In questo modo, quando si sta effettuando un'esecuzione in tempo reale, è possibile, variando il valore di k, ottenere deviazioni temporali più o meno accentuate, pensando così di passare da un'interpretazione in stile barocco (k = 1) ad una in stile romantico (k = 9).

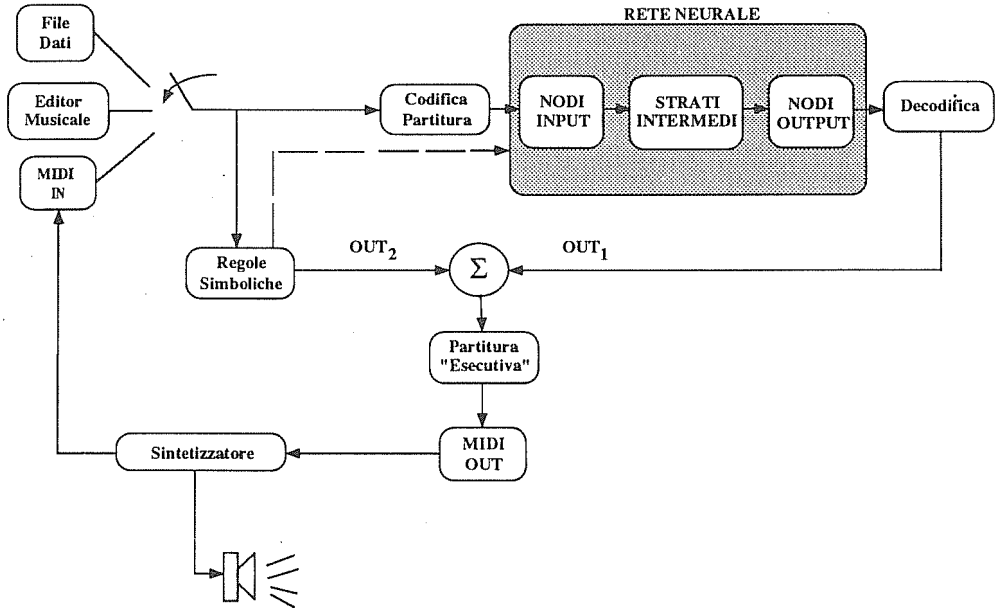


figura 5: Schema del sistema ibrido

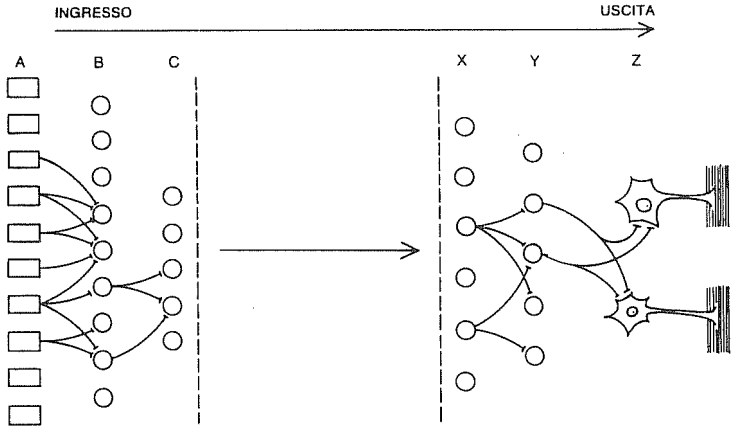


figura 6: Schema del cervello

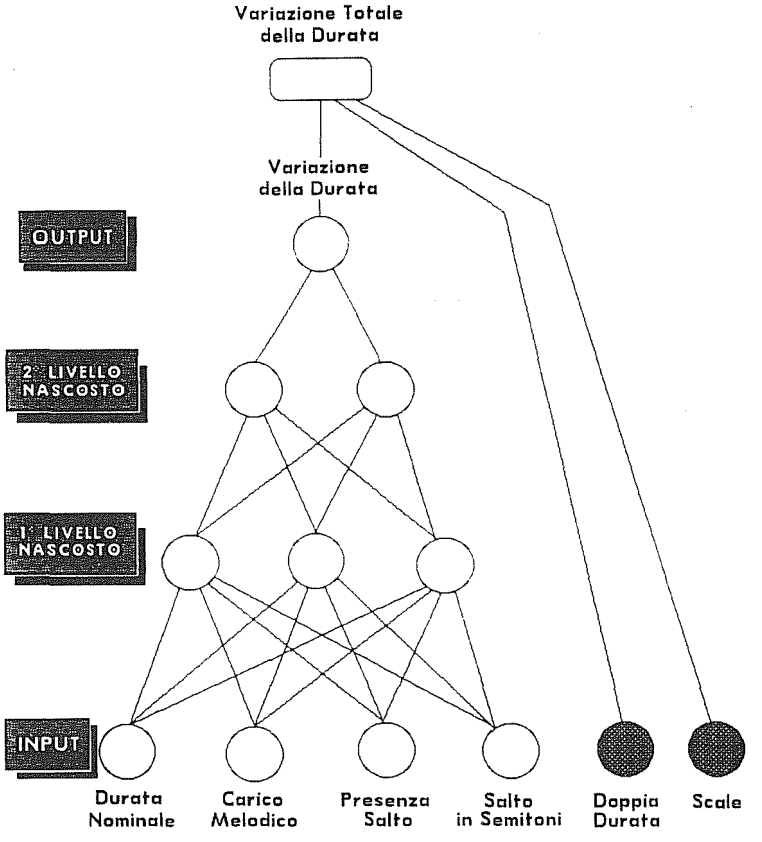


figura 7: Rete ibrida

APPRENDIMENTO DI UNO STILE INTERPRETATIVO.

Un'applicazione molto interessante è quella di riuscire a realizzare una rete neurale in grado di imparare lo stile interpretativo di un pianista affermato, così da poter in seguito eseguire automaticamente delle partiture con il suo stile.

Per l'addestramento si è utilizzato il III movimento della Sonata per pianoforte op. 31 No. 3 di Beethoven. La scelta è caduta su questo brano per eseguire dei confronti con i risultati del lavoro di Repp (1990), basato sulla stessa composizione. In particolare è stata analizzata la melodia di questo pezzo, eseguita al sintetizzatore dal pianista Piero Pittari, il quale ha eseguito più volte la partitura, scegliendo alla fine quelle che egli considerava le migliori esecuzioni. Dall'analisi delle variazioni metronometriche dei singoli quarti di battuta (figure 8 e 9), si nota che un overlap abbastanza buono con i dati riportati da Repp, ottenuti dallo studio delle esecuzioni (su disco o nastro) di diciannove pianisti famosi. Si vedano a tal proposito le figure dalla 8 alla 10. L'andamento a "V" di alcune battute manifesta quello che Clynes chiama il "Beethoven pulse", in cui il II quarto della battuta è più veloce del I e il III quarto è più lento del primo, e quindi

possiamo dire che l'esecuzione a nostra disposizione è abbastanza beethoveniana.

Una volta in possesso di tutti i dati MIDI relativi all'esecuzione di Pittari, si è pensato di realizzare una rete neurale che apprendesse il suo stile interpretativo, con la possibilità, in seguito, di interpretare altre partiture con lo stile di Pittari. Dopo alcune prove condotte utilizzando la stessa rete impiegata per l'apprendimento delle regole di Sundberg e collaboratori, ci si è accorti che tale rete non era sufficiente allo scopo. L'inadeguatezza era dovuta al fatto che erano troppo pochi i parametri in ingresso ai nodi della rete, insufficienti cioè a definire l'interpretazione di un pianista professionista. Si è allora introdotto un ulteriore neurone di input per tener conto dei margini delle legature di frase, in cui si verificano solitamente rallentamenti e/o respiri, codificando se siamo in presenza di un inizio di frase o di un accelerando (valore = 0.1), in un fine di frase o di un rallentando (valore = 0.9) o in nessuno di questi casi (valore = 0.5). I valori scelti si spiegano pensando che dove non ci sia né rallentando, né accelerando non si devono ottenere microvariazioni e quindi si codifica questa situazione con il valore 0.5. Invece il caso di un accelerando, corrispondente ad una deviazione temporale negativa, viene codificato con 0.1 e quello di rallentando corrispondente ad una deviazione positiva con 0.9. Inoltre si sono considerate due reti diverse nel caso si consideri il I o il II ritornello, poichè le deviazioni temporali apportate dal pianista sono decisamente diverse nei due casi, questo per dare maggior "movimento" all'esecuzione. L'addestramento della rete è stato fatto sempre mediante l'algoritmo di back propagation, dove l'output insegnato è la deviazione temporale percentuale della singola nota ottenuta facendo il rapporto tra la durata risultante dall'esecuzione e la durata nominale riportata dalla partitura. Tra le deviazioni percentuali così ottenute sono state scartate quelle con valore troppo elevato, che si sono verificate prevalentemente all'inizio e alla fine del brano e alla fine dei ritornelli.

Il risultato ottenuto è buono: proponendo l'intero minuetto eseguito dalla rete addestrata ad alcuni ascoltatori, questi hanno riconosciuto alcune caratteristiche stilistiche del pianista Pittari.

In figura 10 è rappresentato l'andamento dei quarti delle battute del minuetto eseguito dalla rete addestrata con lo stile del pianista. Si può osservare, che ci sono ancora i caratteristici andamenti a "V" in almeno 20 battute, non sempre corrispondenti a quelli dell'esecuzione originale, ma che comunque danno un andamento abbastanza beethoveniano (secondo Clynes) dell'esecuzione mediante rete neurale addestrata. In ogni caso, a livello microscopico, è importante notare come sia preservato lo stile del pianista (vedi figura 9), dato da deviazioni temporali marcate tra il terzo quarto della battuta e il primo quarto della battuta successiva.

Se confrontiamo questo grafico con quello dell'andamento dei quarti del minuetto eseguito dal sistema ibrido (con K = 9: valore che dà le maggiori deviazioni), si nota che il primo grafico dà decisamente un'esecuzione con deviazioni temporali maggiori. Questo è dovuto soprattutto al fatto che per il sistema ibrido sono state considerate poche regole e quasi tutte di segmentazione e che, in ogni caso, non possono tener conto di quelle implicazioni culturali ed emotive intrinseche nell'esecuzione di un pianista. Per cui i risultati ottenuti sono da considerarsi interessanti.

Per ottenere risultati ancora migliori, per l'apprendimento dello stile di un pianista, si potrebbe utilizzare un modello di rete più sofisticato, come quello della rete neurale ecologica (Parisi, 1990): cioè una rete che tenga conto del fatto che gli stimoli provenienti dall'ambiente (input della rete) nel ciclo N sono funzione delle caratteristiche dell'ambiente ma anche dell'azione della rete stessa (output della rete) nel ciclo N-1. Si potrebbe allora pensare di utilizzare una rete con gli stessi input di quella dell'esperimento precedente con in più un ulteriore nodo di input, che tiene conto dell'output della rete al passo precedente. Si ottiene così una rete neurale con feed-back.

In quest'ultimo caso si può pensare che i neuroni di input rappresentino non solo i nervi ottici, cioè gli occhi del pianista durante la lettura della partitura, ma anche, con il nuovo neurone di input, i nervi acustici, cioè il pianista si ascolta durante l'esecuzione e quindi agisce di conseguenza.

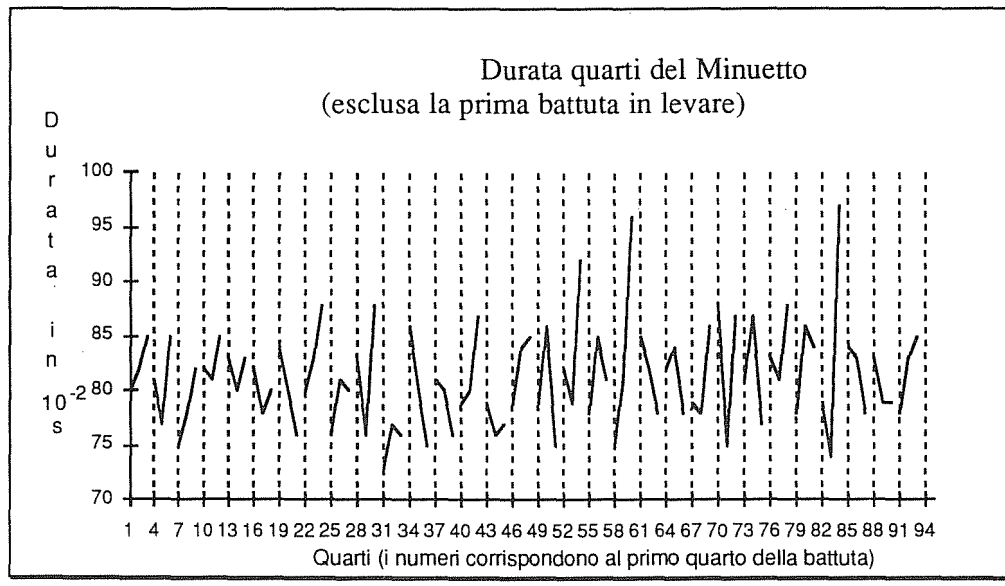


figura 8: Minuetto eseguito dal pianista

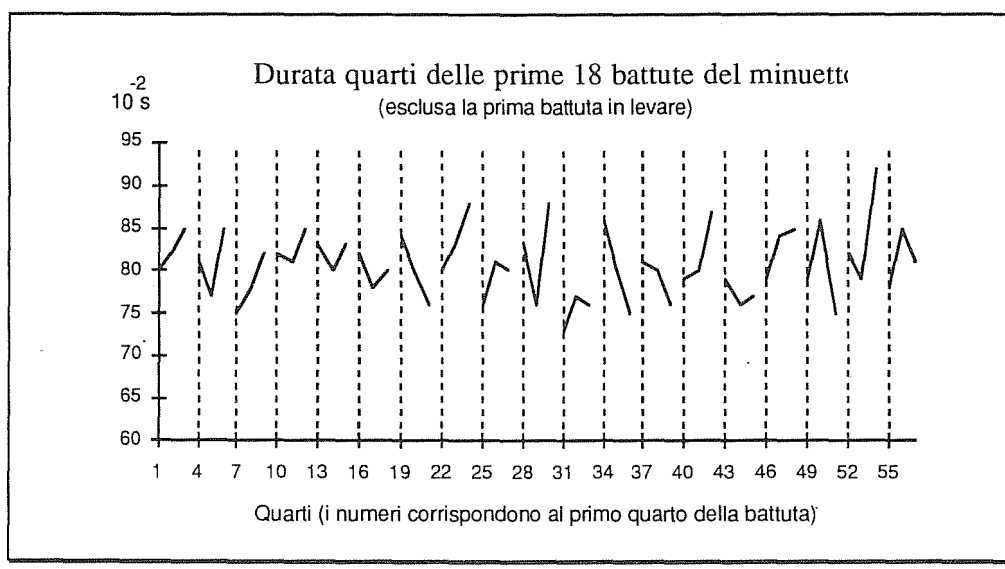


figura 9: Minuetto eseguito dal pianista

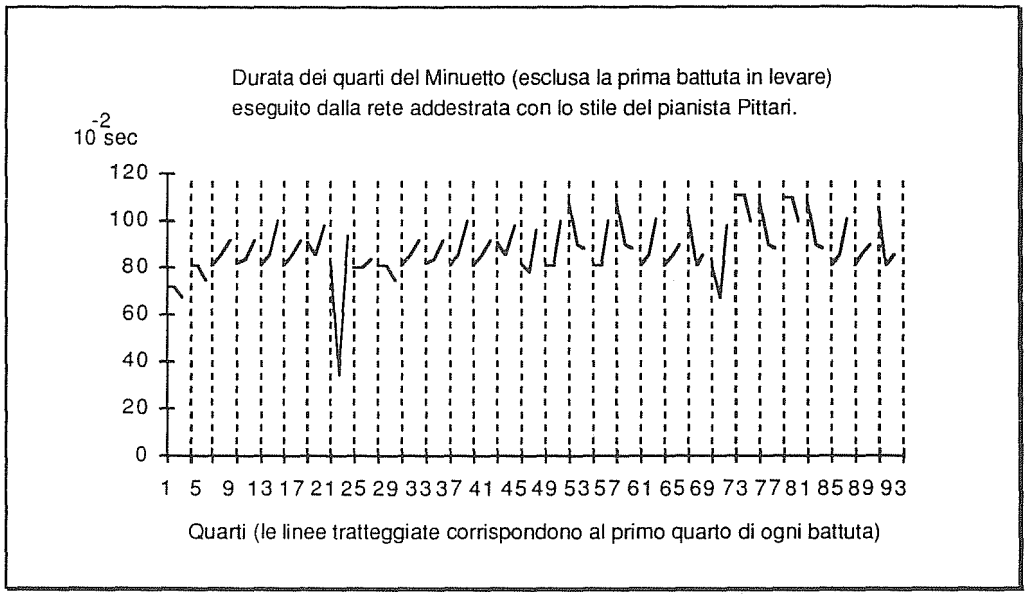


figura 10

CONCLUSIONI

I buoni risultati ottenuti in questo lavoro confermano le ipotesi iniziali riguardo l'efficacia dell'uso delle reti neurali artificiali per l'esecuzione musicale. Tali risultati inducono un ulteriore sviluppo del sistema realizzato. E' importante notare come, con le reti ottenute con la presente ricerca, è possibile eseguire una qualunque partitura in tempo reale, conoscendo solo alcuni parametri fondamentali, che sono la durata nominale della nota, la sua intensità nominale e la sua altezza nel rigo musicale. Questo procedimento è una novità rispetto ai classici approcci per l'esecuzione mediante computer in cui o si vanno a tarare "a mano" i parametri esecutivi delle singole note oppure si usa un sistema esperto, che mediante regole simboliche appronta le caratteristiche espressive di ogni singola nota. Le differenze sostanziali dell'esecuzione mediante reti neurali rispetto ai due metodi citati sono due: la prima consiste nel fatto che con la reti neurali, una volta noti i valori di durata, intensità e altezza di ogni nota, l'esecuzione avviene in tempo reale, la seconda, e più importante, è che l'addestramento di una rete non dipende dal contesto, cioè si possono eseguire indifferentemente, ad esempio, musiche di Beethoven o di Mozart.

Un risultato importante è che, dopo un breve addestramento della rete (circa 15 minuti con un 80386 a 33 MHz) con uno stile particolare, è possibile far eseguire qualsiasi partitura con quello stesso stile ed in tempo reale: man mano che lo spartito viene "letto " dai neuroni di input, esso viene eseguito istantaneamente.

Un'altra conclusione interessante è che si è verificato il previsto "smussamento" delle deviazioni in output rispetto a quelle, esagerate, ottenute tramite l'effetto additivo del sistema di regole simboliche di Sundberg.

L'applicazione più importante di questo lavoro non è tanto l'esecuzione di partiture concepite per strumenti tradizionali, ma di musiche generate da computer, in quanto, per il momento, il computer non è in grado di effettuare autonomamente quelle microvariazioni, di tempo, intensità e timbro, che invece sono le caratteristiche fondamentali che contraddistinguono l'abilità esecutiva di un concertista. Una possibile applicazione del presente lavoro consiste nell'eseguire automaticamente delle partiture generate da computer, introducendo delle microvariazioni sia in

durata che in intensità, rendendo l'esecuzione più "armoniosa" all'orecchio umano.

In particolare sono state prese in considerazione le reti neurali discusse in precedenza che permettono di variare in tempo reale il parametro K delle regole di Sundberg e applicate all'uscita di una sorgente markoviana di melodie (Bresin, Manduchi 1989), che in pratica è un'improvvisazione automatica su un tema memorizzato in precedenza. Variando allora il parametro K durante l'esecuzione, potremo passare in qualunque istante da un'esecuzione "fredda" (bassi valori di K) ad una "romantica" (alti valori di K) a seconda dei gusti dell'operatore.

I risultati ottenuti possono anche avere un'applicazione commerciale, se pensiamo alla possibilità di trasferire le reti nelle ROM dei sintetizzatori, offrendo così la possibilità di esecuzioni automatiche "interpretate".

Con l'uso delle reti neurali si apre quindi una nuova strada nell'esecuzione musicale mediante computer.

RIFERIMENTI

R. Bresin, R. Manduchi "Una sorgente di melodie con controllo di entropia", **Atti VIII Colloquio di Informatica Musicale**, Cagliari 26-28 ottobre 1989, pp. 213-215

G. De Poli, L. Irone, A. Vidolin "Music score interpretation using a multilevel knowledge base", **Interface**, vol. 19, 1990, pp. 137-146

A. Friberg "Generative Rules for Music Performance: A Formal Description of a Rule System", **Computer Music Journal**, vol. 15, No. 2, Summer 1991, pp. 56- 71

A. Gabrielsson "Interplay Between Analysis and Synthesis in Studies of Music Performance and Music Experience", **Music Perception**, vol. 3, No. 1, 1985, pp. 59-86

D.H. Hubel, "Il cervello" , **Le Scienze**, No. 135, Novembre 1979

D. Parisi, F. Cecconi, S. Nolfi "ECONETS: neural networks that learn in an environment", **Networks**, 1, 1990, pp. 149-168

B.H. Repp "Patterns of expressive timing in performances of Beethoven minuett by nineteen famous pianists", **J. Acoust. Soc. Am.**, vol. 88, No. 2, August 1990, pp. 622-641

B.H. Repp "Further Perceptual Evaluations of Pulse Microstructure in Computer Performances of Classical Piano Music", **Music Perception**, vol. 8, No. 1, 1990, pp. 1-33

D.E. Rumelhart, J.L. McClelland **Parallel Distributed Processing**, vol. 1, Cambridge: MIT Press, 1988, chapter 8

D.E. Rumelhart, J.L. McClelland **Explorations in Parallel Distributed Processing**, Cambridge: MIT Press, 1988b, chapter 5, appendix C

J. Sundberg, A. Friberg, L. Frydén "Rules for automated performance of ensemble music", **Contemporary Music Review**, Vol. 3, 1989, pp. 89-109

J. Sundberg et al. "Performance Rules for Computer-Controlled Contemporary Keyboard Music", **Computer Music Journal**, vol. 15, No. 2, Summer 1991, pp. 49-55

J. Sundberg, W.F. Thompson, A. Friberg, L. Frydén "Use of rules for expression in the performance of melodies", **Psychology of music**, vol. 17, No. 1, 1989, pp. 63-82

CLASSIFICAZIONE TIMBRICA PER SINTETIZZATORI MIDI MEDIANTE RETE NEURALE

(Sound Classification for MIDI Synthesizers using Neural Networks)

M.Graziani[#], P.Zollo[+]

[#]C.S.C.: Centro di Sonologia Computazionale
Università di Padova, Via S.Francesco 11, 35121 Padova, Italy
E-mail: music01@unipad.cineca.it
Tel. +39-49-8283711 - Fax +39-49-8754094

[+]Facoltà di Ingegneria Elettronica
Università di Bologna, Italy

Keywords: Neural Networks

ABSTRACT

This paper describes the realization of a neural network to classify the sounds of a MIDI synthesizer.
In the MIDI protocol, the data that control the sound features are part of System Exclusive (SysEx). The SysEx data have a very complex format related to the machine hardware. It is nearly impossible, for an experienced user, to identify the sounds simply looking at SysEx data or at a graphic representation. So, the only way to classify a set of sound data is to hear the sound.
The authors realized a neural network to classify the sounds of an FM synthesizer starting from SysEx data. The synthesizer is the Yamaha TX81Z, an FM machine with 4 operators.
Not all the data are meaningful in the synthesis process. We chose to set apart the parameters that directly control the synthesis algorithm: algorithm number, frequency controls, operator on/off, feed-back, oscillator waveforms, envelope values, output level. Each algorithm is composed by 4 operators so the total number of parameters is 50.

There are 10 instrumental classes: brass, wind, piano, organ, plucked-string, bowed-string, voice, percussion, synth, sound-effect.
We realized two networks to assign each sound to a class.
The first network has 50 input units, 10 hidden and 10 output units. Input data are rescaled between 0 and 1.
In the second, the data are further processed to obtain a binary pattern. In this way, the input pattern is composed by 162 binary digits.
All systems work fine. The last works better, but it is slowed down by the amount of computing.

1. Introduzione

Uno dei maggiori problemi nel campo dei sintetizzatori MIDI consiste nel trattamento dei dati che compongono il cosiddetto "Sistema Esclusivo" (abbr.: SysEx). La questione non è banale in quanto il SysEx contiene, fra gli altri, tutti quei dati che controllano il sistema di sintesi della macchina e determinano, in tal modo, la sonorità effettivamente emessa.
L'insieme dei dati che costituiscono il Sistema Esclusivo, inoltre, è specifico per ogni macchina. Ne consegue che una standardizzazione del software atto a trattare tali dati è impossibile.
L'applicazione delle reti neurali al trattamento dei dati di SysEx è, quindi, desiderabile in quanto potrebbe offrire almeno un approccio metodologico comune ai sistemi delle varie macchine MIDI.

2. Il problema della classificazione

L'estrema diffusione di tali sintetizzatori nella comunità della computer music ha fatto si che venisse sviluppata, in breve tempo, una notevole mole di suoni e una corrispondente quantità di software dedicata alla manipolazione, all'archiviazione e, in alcuni casi, anche alla generazione automatica di dati timbrici per le varie macchine, mediante algoritmi appositamente studiati. Attualmente sono disponibili dei database (i cosiddetti "Librarians") che permettono di archiviare i dati di SysEx per la generazione del suono, suddivisi in categorie strumentali ed etichettati con varie chiavi di ricerca e commenti.
La funzione che, finora, è sfuggita all'automazione è, però, quella della classificazione, che, come in molte altre realtà, costituisce, ormai, il punto più oneroso per l'utente. Con i software attuali, infatti, è proprio l'utente che, se desidera disporre di un archivio ordinato, deve procedere alla classificazione di ogni suono sulla base, almeno, della categoria strumentale in cui può essere inserito.
La natura stessa dei dati di SysEx non è tale da facilitare il lavoro. Questi insiemi di dati, infatti, sono composti da un'alta quantità di parametri: in un sintetizzatore MIDI, il suono è definito da

un numero di valori che varia da un minimo di 50 a circa 200, nelle macchine più complesse. I singoli parametri, inoltre, sono non lineari, interconnessi (l'efficacia acustica di ognuno di essi dipende anche dai valori di altri parametri collegati), espressi in unità di misura strettamente legate all'hardware e non standard. Tale situazione inibisce totalmente il riconoscimento del suono mediante la consultazione diretta dei dati, anche se in forma grafica, da parte di un umano, per quanto esperto: l'unico sistema di classificazione attualmente usato è quello dell'ascolto del suono eseguito dall'utente.
Se si pensa che un utente medio arriva rapidamente a possedere alcune migliaia di suoni per ogni sintetizzatore e che la dotazione normale di uno studio si calcola nelle decine di migliaia, si comprende facilmente l'entità del problema (che provoca anche una sotto-utilizzazione del database adottato).
Un algoritmo in grado di eseguire automaticamente tale passo di classificazione, con minimo margine di errore è, quindi, desiderabile.
Gli autori hanno affrontato il suddetto problema partendo da insiemi di dati timbrici relativi al sintetizzatore Yamaha in modulazione di frequenza TX81Z, avendo a disposizione, su questa macchine, una vasta biblioteca di materiale già classificato da utilizzare nella fase di apprendimento e hanno sviluppato un software, basato su una rete neurale, utile per la classificazione rispetto alla categoria strumentale di appartenenza.

3. Pre-processing

Il TX81Z è ben conosciuto dai musicisti che utilizzano la modulazione di frequenza come sistema di sintesi. Si tratta di una macchina a 4 operatori, ognuno dei quali è composto da un oscillatore e un inviluppo a 4 segmenti. L'oscillatore può utilizzare una fra 8 forme d'onda che vanno dalla sinusoidale a onde con diverso contenuto armonico. Gli operatori vengono connessi a formare 8 possibili algoritmi.
In questo sintetizzatore, ogni suono è totalmente definito da due distinti blocchi di dati: il primo, detto Voice Edit Parameters (VCED), è composto di 93 bytes, mentre il secondo (Voice Edit Additional Parameters: ACED), di 22 bytes. La lunghezza totale del blocco di definizione di una voce è, quindi, di 115 bytes.
Non tutti questi parametri, però, sono significativi dal punto di vista sonoro. Per i nostri fini, essi possono essere suddivisi in quattro sezioni:

1. Valori direttamente collegati alla sonorità e alla sua evoluzione temporale;

2. Valori che controllano determinati aspetti del suono in esecuzione (vibrato, portamento, effetti, ecc.);

3. Controlli di esecuzione ad uso dell'utente, come lo scaling della tastiera, l'after-touch, il pitch-bend, il breath control, eccetera;

4. Valori che hanno nessuna rilevanza sonora o esecutiva, come il nome della voce.

La suddetta suddivisione è ordinata sulla base della rilevanza dei parametri nel problema in esame. I più significativi per la determinazione del risultato sonoro e del riconoscimento del suono collegato sono quelli delle prime due sezioni, mentre la terza assume una certa rilevanza solo in certi suoni. L'importanza dell'ultima, infine, è nulla.
Nel nostro caso, si è deciso di utilizzare, come base della classificazione, solamente i parametri della sezione (1) e di considerare come sviluppo futuro l'inclusione di quelli appartenenti alla seconda.
La seguente tabella riassume i parametri presi in considerazione e il loro significato.

Parametro	Significato	Range
Algoritmo	Tipo di algoritmo utilizzato fra gli 8 possibili	0 - 7
Operator On/off	Controllo di On/Off per ognuno dei 4 operatori	0 - 1
Feedback	Controllo di retroazione in modulazione	0 - 7
Frequency Coarse	Controllo che determina la frequenza dell'operatore in base alla nota selezionata	0 - 63
Frequency Fine	Come sopra, controllo fine	0 - 15
Fixed Frequency On/Off	Switch per attribuire a un operatore una frequenza fissa	0 - 1
Osc. Waveform	Forma d'onda dell'operatore	0 - 7
Attack Rate	Velocità dell'attacco nell'inviluppo di ampiezza	0 - 31
1st Decay Rate	Velocità del primo segmento di decadimento	0 - 31
1st Decay Level	Livello raggiunto con il primo decadimento	0 - 15
2nd Decay Rate	Velocità del secondo decadimento	0 - 31
Release Rate	Velocità del release	1 - 15
EG Shift	Livello iniziale e finale dell'inviluppo cioè ampiezza minima	0 - 3
Output Level	Livello di uscita cioè ampiezza massima	0 - 99

Si rende, quindi, necessario un certo lavoro di pre-processing per l'isolamento dei parametri prescelti, anche considerando che parte di essi è codificata o compressa in forma legata all'hardware.

É stato, quindi, realizzato un apposito software che accede alle librerie di suoni e si incarica di estrarre i singoli presets isolando i valori prescelti.
Considerando che esistono 4 operatori, i parametri utilizzati sono 50.

4. Classificazione

Ogni timbro, quindi, è visto come una configurazione di 50 valori che deve essere classificata in base alle 10 categorie strumentali di uscita.
Le categorie utilizzate sono quelle tipiche degli Editors/Librarians in uso nel MIDI e cioè:

1. Brass
2. Wind
3. Piano
4. Organ
5. Plucked String
6. Bowed String
7. Voice
8. Percussion
9. Synth
10. Sound Effect

Si desidera, cioè, disporre di una rete nella quale ogni configurazione di input attivi la configurazione di output corrispondente.
Allo scopo di verificare l'applicazione delle reti neurali a questo tipo di problema, minimizzando la quantità di calcoli necessari, sono state effettuate ricerche con diversi tipi di configurazioni, pur restando entro il comune modello di rete forward con uno strato di input, uno di unità nascoste e uno di output.
Un primo modello prevede l'input di dati numerici continui fra 0 e 1.
Per generare una corretta configurazione di input, i valori dei parametri sonori precedentemente isolati vengono, poi, riscalati fra 0 e 1 tenendo conto dell'effettivo andamento fisico del parametro collegato. Mentre alcuni valori, come quello dell'algoritmo, sono puri

numeri, altri hanno un corrispondente fisico che non segue necessariamente l'andamento del valore numerico.
Il comportamento di ogni parametro è determinato dalle modalità costruttive del sintetizzatore in esame. Un riscalamento e riadattamento dei valori per ottenere i corrispondenti fisici risulta necessario.
Il livello di uscita, per esempio, pur essendo calibrato numericamente fra 0 e 99, non varia linearmente, ma secondo una curva esponenziale. I tempi di attacco e decadimento non sono proporzionali ai valori numerici, ma cambiano secondo una tabella ben precisa. Il valore di Frequency, che è critico in quanto rappresenta i rapporti fra portanti e modulanti all'interno dell'algoritmo e determina, quindi, quali componenti sono presenti nello spettro del suono in uscita, si ricava combinando, secondo particolari tabelle, i parametri Frequency Coarse e Frequency Fine, ricavandone un valore che va da 0.50 a 27.50.
La rete utilizzata, quindi, è composta di 50 unità di input, 10 di output e uno strato di 10 unità nascoste.
In tale modalità, è desiderabile che ogni configurazione di input attivi una e una sola unità di output.
Una variazione di tale modello prevede la riduzione delle unità di output al numero di 4 essendo questo il minimo quantitativo necessario per selezionare una delle 10 categorie possibili. Ogni configurazione di input, quindi, deve produrre una configurazione delle 4 unità di output.
In un secondo modello, invece, il pre-processing tende a trasformare i singoli parametri in configurazioni binarie con un certo numero di bit per ogni parametro. L'attacco, per esempio, viene classificato, in base alla durata, con un valore da 0 a 7 ed è codificato in una stringa di 3 bit. In modo analogo si procede per gli altri parametri. Il suono viene, così, visto come una stringa binaria di 162 elementi che vengono avviate ad altrettante unità di input della rete.
Entrambi i metodi producono risultati apprezzabili. Il secondo, in particolare, produce risultati migliori, ma è penalizzato da una maggiore quantità di calcoli.
L'apprendimento è stato condotto su un set di 50 configurazioni, utilizzando la classica back-propagation, con software realizzato dagli autori su sistema MS-DOS 386, 4Mbyte RAM.

5. Sviluppi attuali

L'attuale linea di sviluppo è diretta a verificare le possibilità di riconoscimento della rete aumentando il numero delle unità di uscita (e conseguentemente, il set di apprendimento), per poter lavorare con una suddivisione più raffinata in grado, per esempio, di distinguere fra i vari tipi di "keyboard", "brass" e "wind", in modo da migliorare le capacità di classificazione.
Un problema che si sta affrontando, inoltre, è quello della classificazione di sonorità che includono più tipologie sonore diverse, entrambe acusticamente evidenti (es.: attacco tipo percussione con coda di suono di flauto), che possono essere realizzate mediante questi sintetizzatori.

DUE RETI CON MEMORIA A BREVE TERMINE PER LA GENERAZIONE DI SEQUENZE ARMONICHE E DI PATTERNS RITMICI IN TEMPO REALE

(Two Short-Term Memory Networks to Generate Harmonic Sequences & Rhythmic Patterns in Real-Time)

M.Graziani[#], P.Zollo[+]

[#]C.S.C.: Centro di Sonologia Computazionale
Università di Padova, Via S.Francesco 11, 35121 Padova, Italy
E-mail: music01@unipad.cineca.it
Tel. +39-49-8283711 - Fax +39-49-8754094

[+]Facoltà di Ingegneria Elettronica
Università di Bologna

Keywords: Neural Networks

ABSTRACT

This paper describes two neural networks to perform music in a real-time environment.
The first is a network to find out the chords on a bass line. This network has 7 input units, an hidden layer and 7 out units. Each input and output unit represent a scale degree. The performer plays a single note bass line activating the corresponding input unit and the network activates the output unit which represents the corresponding harmonic chord.
Since the order of events is important we used a Jordan network. This kind of network has a copy of output units at input level. So the input units are doubled. The output pattern is copied in this units so the network has a memory of every preceeding event.
A similar network can be used to generate rhythmic patterns composed by 8 or 16 steps to represent quavers or semiquavers. An input unit is activated if the user plays a note on the corresponding quaver or if the user plays an accented note. The total input configuration is a bar long.
The network we used has a copy of the last input configuration at the input level, so the machine can remember the last bar user played. The network generates a response pattern by activating the output units.

Introduzione

Gli autori hanno studiato due piccole reti, dotate di memoria a breve termine, utili nella programmazione di ambienti esecutivi in tempo reale o di strumenti intelligenti.
La prima effettua un riconoscimento di carattere armonico determinando il grado della scala su cui costruire l'accordo, in base al grado della nota utilizzata nella linea del basso. La seconda genera un pattern ritmico basandosi sul pattern eseguito dall'utente.
Le varie esperienze nella programmazione di ambienti esecutivi per il tempo reale, compiute, fra gli altri, da uno degli autori, che ha lavorato dal 1983 al 1989 sul Sistema 4I presso il C.S.C., hanno evidenziato la notevole complessità di definizione delle modalità di risposta della macchina alle azioni dell'esecutore.
Tali difficoltà sono evidenti soprattutto se si considera che, in un ambiente esecutivo o in uno strumento intelligente, la macchina non può limitarsi alla pura e semplice esecuzione, ma deve effettuare qualche tipo di riconoscimento "intelligente" delle azioni del musicista per generare anche i dati relativi agli eventi di contorno e a quelli di basso livello.
Alcune caratteristiche delle reti neurali le rendono particolarmente adatte all'impiego in questo contesto. In particolare, esse

- sono configurabili: per quanto lunga possa essere la fase di apprendimento, la ri-configurazione non comporta un cambiamento dell'ambiente a livello di progettazione;
- hanno un elevato grado di tolleranza all'errore e quindi sopportano bene le imprecisioni dell'esecutore umano;
- possono essere dotate facilmente di memoria a breve termine;
- una volta configurate, hanno un tempo computazionale fisso;
- mantenendo le loro dimensioni entro limiti ridotti, possono essere utilizzate anche in tempo reale.

Spinti da queste considerazioni, gli autori hanno iniziato a studiare l'impiego delle reti neurali nella programmazione di ambienti esecutivi in tempo reale o di strumenti intelligenti, sviluppando due piccole reti.
La prima effettua un riconoscimento di carattere armonico, determinando il grado della scala su cui costruire l'accordo, in base al grado della nota utilizzata nella linea del basso. La seconda genera un pattern ritmico, di lunghezza pari a 8 o 16 steps, basandosi sul pattern eseguito dall'utente.

Riconoscimento armonico

Nell'armonia tradizionale, è sufficiente conoscere la tonalità (con relativa scala) e le note utilizzate, via via, nella linea di basso per costruire una serie di accordi.
Il processo può essere ulteriormente semplificato, considerando che, lavorando solo sui gradi della scala, il tutto diventa anche indipendente dalla tonalità. L'idea, quindi, è quella di fornire alla rete la nota del basso, espressa come grado della scala, per ottenere, in uscita, la determinazione dell'accordo, espresso come grado della scala su cui costruirlo. Tale costruzione, se non si tiene conto del rivolto, è, poi, puramente meccanica.
Se non si tiene conto della sequenza, per eseguire questo compito basta una rete con 7 unità di input, uno strato intermedio e 7 unità di output.
La difficoltà, però, è costituita dal fatto che la sequenza è importante. Per questa ragione, abbiamo utilizzato una rete di Jordan con 7 unità di stato sulle unità di output, arrivando a un totale di 14 unità di input.
L'input della rete è, così, costituito dal corrente grado della nota del basso più il grado su cui è stato costruito l'ultimo accordo utilizzato. In tal modo, si realizza una rete forward in cui si può utilizzare la back-propagation, ma che, nello stesso tempo, conserva una traccia degli ultimi N outputs.
Nonostante vi sia una retroazione, è ancora possibile utilizzare la back-propagation per l'apprendimento, in quanto la retroazione non è veramente tale. Il pattern di output, infatti, viene copiato pari pari nelle unità di stato senza che vengano tenuti in considerazione dei pesi che non devono, quindi, essere cambiati nel corso dell'apprendimento. Quello che accade è semplicemente il fatto che la configurazione di input è data dall'accostamento del pattern di input corrente e dell'ultimo pattern di output.

Generazione di Patterns ritmici

Una rete a 3 strati, analoga alla precedente, può essere utilizzata anche per generare patterns ritmici in risposta a quelli forniti dall'utente. Il pattern in input è costituito da un certo numero di unità, tipicamente 8 o 16 per rappresentare, per esempio, una battuta suddivisa in crome o semicrome. Le possibili modalità di attivazione delle unità di input sono due:

- l'unità è attivata solo se l'esecutore ha piazzato un suono sulla croma o semicroma corrispondente;
- l'unità è attivata se l'esecutore ha posto un accento dinamico sulla croma o semicroma corrispondente.

In entrambi i casi, il pattern in input può essere visto come una serie di valori 0/1 in cui l'uno rappresenta la presenza di un suono/accento. La rete provvede a generare un pattern rappresentato dagli stati delle unità di output, il cui numero, di solito, è pari a quello delle unità di input (ma stiamo sperimentando anche configurazioni diverse).
In questo caso, però, le unità di stato sono poste sulle unità di input anziché su quelle di output. La rete mantiene, così, una limitata memoria dell'ultimo pattern eseguito dall'utente.
Uno sviluppo interessante, su cui si sta concentrando la nostra attenzione, si ottiene aggiungendo anche delle unità di stato che reimmettono in input l'ultimo pattern di output. In tal modo, la rete è sensibile anche ai propri patterns generati nel passato prossimo.

APPLICAZIONE DELLE RETI NEURALI ALLA CLASSIFICAZIONE DEI REGISTRI DELL'ORGANO A CANNE.

R. Bresin, G. De Poli, G. Torelli

C.S.C. - D.E.I.
Università degli Studi di Padova
via Gradenigo 6/a, I - 35131 Padova
E.mail: ADTPOLI@ipduniv.bitnet
Tel. +39 (0)49 8287631 - Fax +39 (0)49 8287699

ABSTRACT

In this paper a method for a organ pipe timbre classification is proposed.

This method consists in teaching neural networks (using the back-propagation algorithm) to classify a timbre in input to the net and giving an output representing a phase in a polar diagram, according with the method proposed by Pagdham (Pagdham, 1986).

INTRODUZIONE

Il problema del riconoscimento del timbro presenta degli aspetti sia oggettivi che soggettivi ed è quindi di difficile soluzione. In questo ambito abbiamo considerato i dati forniti da Pagdham (Pagdham, 1986), il quale ha proposto una classificazione dei registri dell'organo a canne mediante dei gruppi di ascolto. Debiasi e Dal Sasso (Debiasi & Dal Sasso, 1989) hanno cercato di automatizzare la classificazione dei registri con l'uso dell'algoritmo proposto da Pagdham. Noi abbiamo pensato di utilizzare le reti neuronali (RN) in quanto hanno un funzionamento non lineare e bene si adattano a problemi di percezione ed organizzazione.

In questo lavoro ci siamo proposti di valutare l'efficacia dell'applicazione di alcuni modelli di RN al problema del riconoscimento ed eventuale classificazione dei registri dell'organo a canne, a partire dalla configurazione spettrale dei suoni.

Il tipo di apprendimento utilizzato per le RN è stato quello mediante back-propagation (Rumelhart & McClelland, 1988) e come rete si è scelta quella con 1 livello di input, 1 livello nascosto e 1 neurone di output.

Per quanto riguarda il criterio di classificazione ci si è riferiti a quello proposto da Pagdham: i timbri dei registri degli organi a canne sono riportati in un grafico polare dove la fase rappresenta il timbro e il modulo la complessità del suono, mentre i 4 assi cartesiani rappresentano le principali famiglie di registri base (flauti, principali, viole e ance).

ALCUNI ESPERIMENTI ESEGUITI

Si sono seguite varie strade. Nella prima si sono utilizzati come dati quelli riportati nel lavoro di Debiasi e Dal Sasso. La configurazione scelta per la RN è stata la 6+5+1 (6 nodi di input, 5 nodi nel livello intermedio, 1 nodo di output). Si sono adottati 6 neuroni di input poiché i dati a disposizione fornivano i valori della fondamentale e delle successive 5 armoniche (normalizzati rispetto al valore efficace). La scelta dei 5 nodi intermedi è giustificata dal fatto che Dal Sasso ipotizzava che la fondamentale non avesse grande peso nella valutazione del timbro, ma per non escludere questa informazione fin dall'inizio si è scelto di ridurre il numero di nodi a 5 solo nel livello intermedio. Come output la rete dà la fase che identifica il timbro nel diagramma polare secondo il metodo di Pagdham. L'addestramento è stato eseguito con 20 timbri scelti in modo da addestrare la rete con dati che coprissero, compatibilmente con quelli disponibili, tutta la gamma di timbri dell'organo (5 flauti, 3 bordoni, 5 principali, 1 viola, 6 ance) e per il test si sono utilizzati i restanti 19 a disposizione: di essi 14

sono stati riconosciuti correttamente. In tabella 1 sono illustrati i risultati per singola categoria.

FLAUTI:	1 su 3 correttamente classificati
	2 classificati come Bordoni
BORDONI:	0 su 1 correttamente classificati
	1 classificati come "iperflauti"
PRINCIPALI:	4 su 4 correttamente classificati
VIOLE:	<assenti>
ANCE:	9 su 11 correttamente classificati
	1 come Viole
	1 come Principali

Tabella 1

I risultati sono stati certamente buoni sia per le Ance che per i Principali, per i Bordoni non si possono trarre conclusioni per scarsità di suoni di test, e per i flauti pare che i risultati non siano molto buoni ma anche in questo caso avendo avuto a disposizione più suoni si avrebbe potuto dire qualcosa di più. Nulla si può dire per le Viole, visto che l'unico suono a disposizione è stato impiegato per l'addestramento.

Sempre con gli stessi dati si è addestrata una rete 6+3+1 per cercare di estrarre dall'analisi dei suoi pesi delle regole di carattere generale. Come pattern di test sono stati utilizzati gli stessi 19 del caso precedente di cui 13 sono stati correttamente classificati. Si veda la tabella 2 per i risultati.

FLAUTI:	2 su 3 correttamente classificati
	1 classificati come Bordoni
BORDONI:	0 su 1 correttamente classificati
	1 classificati come "iperflauti"
PRINCIPALI:	4 su 4 correttamente classificati
VIOLE:	<assenti>
ANCE:	7 su 11 correttamente classificati
	3 come Viole
	1 come Bordoni

Tabella 2

Nella seconda serie di esperimenti abbiamo considerato i dati riportati nell'articolo di Pagdham. Il tipo di rete utilizzato è stato il 5+5+1, poiché si è esclusa la fondamentale e si sono prese come input solamente le prime 5 armoniche successive alla fondamentale. I valori a disposizione erano in dB e normalizzati rispetto alla fondamentale. L'addestramento è stato condotto con 20 pattern ed il test con altri 60, dei quali 46 sono caduti dentro la banda $[-3\sigma,+3\sigma]$, dove σ è la deviazione standard. Anche questa volta utilizzando una rete 5+3+1 i risultati sono stati qualitativamente simili: 41 suoni su 60 dentro la banda $[-3\sigma,+3\sigma]$.

Infine si confronteranno i risultati ottenuti con quelli ricavati con gli altri approcci e si discuteranno le prospettive dell'impiego delle RN nella classificazione dei timbri.

RIFERIMENTI

G.B. Debiasi, M. Dal Sasso "Metodo per la valutazione automatica dei timbri di organi a canne", **Atti VIII CIM**, Cagliari 1989, pp. 17 - 25

C. Pagdham "The scaling of the timbre of organ pipe", **Acustica**, Vol. 60, No.3, Mai 1986, pp. 189 - 204

D.E. Rumelhart, J.L. McClelland **Parallel Distributed Processing**, vol. 1, Cambridge: MIT Press, 1988, chapter 8

Capitolo 2: Intelligenza Artificiale e Scienze Cognitive

THE SCHEMA CONCEPT - A CRITICAL REVIEW OF ITS DEVELOPMENT AND CURRENT USE IN COGNITIVE SCIENCE AND RESEARCH ON MUSIC PERCEPTION

U. Seifert

Department of Musicology & Department of Computer Science
University of Hamburg, Neue Rabenstr. 13, W-2000 Hamburg 36, Germany
E-mail: fk00010@dhhuni4.bitnet

Keywords: cognitive science, cognitive musicology, neuromusicology, psychomusicology, music perception, connectionism, schema, cell assembly, modules, columns, blackboard-architecture.

Abstract: A classification of current research on music influenced by computer science is given. Musical Informatics, computer music and cognitive musicology, i. e. "music and cognitive science" are distinguished. Cognitive musicology is subdivided in structural or theoretical musicology, psychomusicology (the information psychological approach in cognitive music psychology), and neuromusicology. It is shown that the schema concept is central to relate these different levels of research in cognitive musicology, for that reason a review of the idea of schemata concerning human cognition is given. The central common features in the ideas on schemata in such different disciplines as philosophy (Kant), neurology (Head), psychology (Bartlett, Piaget, Neisser), neuropsychology (Hebb), artificial intelligence (Minsky) and cognitive science (Arbib) are pointed out. A frame of reference is established which makes it possible to relate the central ideas on schemata in different disciplines. Therefore a vertical and horizontal stratification of the contents of consciousness is given. With respect to verticality there are acts and phenomena to be distinguished. This distinction allows to relate the goals of research in cognitive psychology as defined by Neisser to the psychology of act which is central to the thinking of the german musicologist, psychologist and philosopher Carl Stumpf. Horizontally, sensation, perception and cognition are distiguished which are, on the other hand, related through schemata and the Hebbian psychological theory. The main aim of this paper is to relate conceptually research in psychomusicology and neuromusicology. The idea of schema is central to connect the work in artificial intelligence, psychology and neuroscience. It is pointed out that in cognitive modeling of higher brain functions it is important to model processes (functions, acts) and not the relations between phenomena. Contrary to current developments in cognitive psychology and artificial intelligence, it is argued that associtive connectionist modeling with randomly connected networks of higher brain functions is implausible, as indicated by current neurophysiological and neuroanatomic research and on psychological grounds. The neurological concept of a functionally and morhpologically structured group of neurons - the module or column concept - as basic processing unit of the cerebral cortex is related to the basic functional unit of psychological analysis, the schema. It is further shown how the schema concept can be interpreted in terms of concepts from artificial intelligence, especially knowledge sources in blackboard architectures. It is men-

tioned that to solve problems like learning/induction and debugging are, for AI, the basic requirements for being able to deliver an adequate contribution to psychological and music theoretical modeling of music perception.

1. Introduction

Nowadays three main areas of research in the field of computers and music are to be distinguished: musical informatics, computer music and cognitive science and music, the latter also called cognitive musicology.
In musical informatics the main goal is to develop computer music languages, man-machine interfaces etc. For short, it is the technological side of the field computers and music.
Artistry one is finding in computer music, the field of artistical activity in creating music - composed or played - via computer.
Scientific research in music - i. e. music theory and music perception - takes place in musicology or, if one is working in the research program of cognitive science in which the "computer metaphor" plays an important role, one is talking of cognitive musicology.
In cognitive musicology we find three main levels of research: music theory or structural musicology, psychomusicology and neuromusicology (for a more detailed discussion of these and related concepts like cognitive psychology of music, AI and music etc. see Seifert 1990).
From the view point of traditional musicology, music theory is the most important area of research, but in cognitive science the goal is to characterize as formal as possible the underlying representational and procedural capacities of a cognitive system in a specific cognitive domain. Thus in cognitve musicology it is necessary to relate music theoretical research with the research on music perception in psychomusicology on one side. On the other side it is necessary to relate psychomusicology and neuromusicology. One conceptual link between these areas is the concept of a "schema", the basic notion of "schema theory", which was also proposed as a unifying concept for research in cognitive science. In this paper I will focus on the relation between psychomusicology and neuromusicology.
In cognitive psychology, especially the cognitive psychology of music, there is a revival, influenced by artificial intelligence, of the old idea of "schemata".
In cognitive psychology of music, "schema" are very often used as representational models for musical structure in the mind or brain especially as memory or knowledge structure and are also used to explain learning, selective attention in

perception etc. (e. g. Bharucha 1985, pp. 128; Bruhn 1988; Carterette & Kendall 1989; Kessler, Hansen & Shephard 1984; Nauck-Boerner 1988; Pressing 1988, p. 133; Stoffer 1985)
To give just one example of the use of "schema" in current research let me quote a passage from a recent review on human music perception (Carterette & Kendall 1989, pp. 144): "*A musical schema can be as particular as knowing the rhythmic structure of a waltz or as general as knowing the fused scalar and melodic properties of the ragas of North Indian music, or of Western polyphonic music.*" The gestaltist Kurt Koffka (1935) was the first psychologist who interpreted musical structures in terms of schemata and the musicologist E. T. Cone (1968) applied the idea of schemata to music theory in the first chapter of his book "Musical form and musical performance". There is no doubt that the idea of schemata is a powerful tool to relate different areas of research on music such as psychology, neurology, artificial intelligence and music theory.
But before tracing the development of the schema-idea back through philosophy, neurology, psychology, and artificial intelligence it is necessary to develop a conceptual framework in which it is possible to relate psychology and neurology, especially when one is trying to connect research in psychomusicology and neuromusicology.

2. A Preliminary Classification of Mental Objects Relevant to Cognitive Science

In psychology there is no clear distinction between the terms; e. g. Hebb (1975, pp. 267) distinguishes sensation from perception in neurological terms. For him, sensation is neural activity up to the dieencephalon in the endbrain with no content in consciousness evoked by an external or internal stimulus. Perception is the neural activity of parts in the cortex cerebri and so building the content of consciousness determined by a stimulus. For Helmut Frank (1969) "*perception*" is like the Hebbian "*sensation*" and Hebb's concept of "*perception*" he calls "apperception". Speaking of cognition, this term often is used to cover phenoma as perception, learning, categorization, thinking. Therefore I have to be explicit about the terminology I use in this article. It is best to start with an introspective classification of the content of consciousness. Let me call the contents of consciousness in general *mental objects* (Changeux 1984, p. 173 & p. 179). A little bit of introspection will show that the have to distinguish between vertically and horizontally stratified mental objects.
Vertically one can distinguish between *phenomena* and *acts*, or as the logician, philosoph, psychologist and musicologist Carl Stumpf (1906) did, between "Erscheinung" (appearence) and "Funktion" (function). In the following,I will use his terminology. The distinction of "appearence" and "functions" as the main classes of mental objects is on the basis of the so called psychology of act (Brentano, Stumpf, Husserl) and is in strong connection to the concept of intentionality, which was developed by Brentano to distinguish the psychical from the physical. To give you a clear idea of this distinction, think of a person who is listening to music. In listening to music you can distinguish the act of listening from the phenomena, the sounds you hear. When you hear, you hear something, but you can recognize that you are hearing. In general: In perception you can distinguish the *percept* and *perceiving*. For example in psychoacoustics one is interested in the relation of stimulus and percept and not in the function of perceiving. For recognizing this distinction it is relevant to see how the research

in cognitive psychology as Ulric Neisser (1967) defined it and so how research in psychomusicology differs from research in psychophysically orientated music research following Helmholtz like research on pitch structure, timbre, scales, temperament (see Carterette & Kendall 1989 pp. 133-144).
Horizontally one can distinguish sensation, perception, and cognition with knowledge as mental objects. No clearcut distinction can be made. This can count as an advantage, because here we have interfaces to relate the different levels and, normally, in listening to music all three levels are involved.
Let me call the phenoma of sensation *primary percepts*, the phenomena of perception *secondary percepts* and the ones of cognition *concepts* (Changeux 1984, pp. 179). To distinguish between these concepts, let me briefly indicate the idea behind this distinction. Sensory physiology and psychophysics are concerned with primary percepts. The attributes of primary percepts are duration in time, localisation in space, quality and intensity (Hensel 1970, pp. 362), e. g. think of a primary tonepercept (a sensation) evoked by a sinuswave (e. g. Keidel 1975; Terhardt 1986 and related work).
Out of primary perceptions the secondary perception is formed, in which gestalt phenomena play an important role; Carl Stumpf (1906) is talking about "Gebilde". Think of tone sequences recognized as melodies.
With cognition we are entering the level of knowledge and naming. Hebb (1975, 95-96) characterizes cognition as perception dominated by thinking and knowledge, which is based on "mediating processes". He (Hebb 1975, p. 40 & p. 169) defined knowledge as a kind of perception, whose results are resisting for some time. It is evident that in listening to music all levels are involved at the same time and that the different levels could only be analyzed after a kind of holistic process.
Before discussing the relevant ideas concerning the schema concept let me summarize the essential results of the last paragraphs. There are two main classes of mental objects: appearences und functions. Cognitive psychology is concerned with research on functions.
Horizontally one can regard primary percepts, secondary percepts and concepts. There is no clearcut border and one can regard secondary percepts as built hierachically of primary percepts and cognition as a special form of perception involving knowledge.

3. Schemata and Philosophy

In philosophy Kant was the first writer who systematically used the concept of schema. His main problem in epistemology was to determine the conditions and limits of human knowledge, and to relate the work of british empiricists like Hume and Locke with the rationalist tradition of Descartes and Leibniz. To characterize his central problem I give the position of the empiricist Locke which could be characterized by the proposition: *Nihil est in intellectu, quod non prius fuerit in sensu*, to which Leibniz objected: *Nisi intellectus ipse*. To answer one of his main questions concerning how conceptual knowledge and sensory data are related he used the idea of schema.
In general Kant used the term "schema" to denote a procedure by which concepts are related to sensory data which are independent of the mind. Kant (1781, A 141; see also B 180): "*In der That liegen unsern reinen sinnlichen Begriffen nicht Bilder der Gegenstände, sondern Schemate zum Grunde.*" (The fact

is that our pure sensuous concepts do not depend on images of objects, but on schemata). The "transcendendal schema" mediates between the universals (Allgemeinbegriffe) - the Kantian categories - and the special entities of sensory perception. To put it briefly in psychological context: the Kantian "schema" is a procedure which connects perception and cognition. Kant was concerned with the foundational patterns of common sense perception and his reflections on the problem of how concepts bring unity into experience could be viewed as prelimanary studies which must be at the basis of every epistemology (Lohmar 1991, p. 77). But how this is done exactly still remains an open question.
The sensory physiologist Irjoe Reenpää (1962) attacked the Kantian problem from the viewpoint of modern logic and general sensory physiology, but this should not be followed up in this context.

4. The Origin of the Schema Concept in Clinical Neurology

Henry Head used the term schema to explicate posture, the phenomenon of phantom limbs and psychic insults connected with brain damage like in aphasia. His view of mental and neural activity was strong influenced by the british physician and neurologist Hughlings Jackson. In their thought there is a strict distinction between the "physical" and the "psychical", and to confuse these two categories in talking about mental deseases which are connected with brain damage for them is a categorical mistake (Head 1920, pp. 49; see also Mackay 1984). The London neurologist Henry Head contradicted the locationalist position in brain research, which dominated brain research at the end of the ninetenth century and is still dominating now, and tried to locate function in neural structure and brain regions. The mistake that the locationists made was to confuse the physiological brain center involved in the destruction of a mental function with the localisation of that function, as Hughlings Jackson said "*To locate the damage which destroys speech and to localise speech are two different things*" (Head 1926, p. 138). Head (1926, 436) wrote: "*The deeper and* wider therefore the injury to the cortex and the underlying structures, the grave*r and more permanent is likely to be the loss of function; but we must never forget that it has disturbed a highly organized act and has not removed a strictly definable anatomical "centre".*"

For Head the destruction of an ordered sequence of physiological processes in neural tissue results in an destruction of mental function. These "highly organized act" he termed "schema or schemata" (Head 1920, p. 605 & p. 722; Head 1926, p. 435, pp. 488 & 533; Head & Holmes 1911, 186). In connection with the sensory cortex Head (1926, p. 435; see also Head 1920, 607) writes: *"But in addition for its function as an organ of local attention, the "sensory" cortex registers and retains the physiological dispositions produced by past events. These profoundly modify all subsequent actions. They may be manifest in the form assumed by sensations or images, but more often, as in the case of special impressions, remain outside consciousness. Her they form organized models of ourselves, which the termed "schemata". Such schemata, although they may act solely on the physiological level, are essential for all accurate spacial recognition; for they modify the impressions produced by incoming sensory impulses in such a way that the final sensations are charged with those a relation to something that has happened before."* We see that for Henry Head schemata are the building blocks for mental activity, working at an unconscious level, materially

building up sequences of neurophysiological activity in the neural tissue and functionally connecting past experience with the present impulses.
Summary: The schema therefore is a physiological concept of undefined structure and not specified activity of the neural tissue of the cortex. Many of these "unconscious" (vorbewußte) activities of schemata are related to conscious experience and they determine conscious experience.. They are important for memory and remembering. Schemata are relevant to mental *functions* and they are not localized in any "brain center" of mental function. The term "center" should be replaced by "preferred center of integration".

5. Schemata and the Psychology of Remembering

The "schemata" as a central idea in thinking about memory and remembering was exploited by F. C. Bartlett (1932). Bartlett, a student of Head, used the term schemata to think about higher mental processes in relation to the physiological level. Bartlett (1926; 1932, pp. 199) is critizising some aspect in Head's definition of schemata. In his "Studies of Neurology" Head (1920, 607) is characterizing the memory function of schemata in terms of a "store-house". This gives the impression that remembering is passive and that things are put somethere and found without changing their structure if they are needed. Bartlett (1932, 200) is critizing this view and points out that memory, storing and remembering are active processes which change the stored content as indicated by his experimental studies on remembering.
Because the term "schema" for Bartlett (1932, 201) is at the same time too sketchy and too definite he proposes to think of schemata as "*active, developing patterns*" and calls them "*organized settings*". But he continues to use the term schema. He (1932, 201) extended the use of "schemata" from mere perception or low level mental activities to high level activities. Bartlett (1932, 201): "*All incoming impulses of a certain kind, or mode, go together to build up an active, organised setting: visual, auditory, various types of cutaneous impulses and the like, at a relatively low level; all the experiences connected by a common interest: in sport, in literature, history, art, science, philosophy and so on, on a high level.*"
Here we see the problem in connecting physiological thinking with psychological thinking. Bartlett is speaking of "impulses" (a neurological concept) and experiences (a psychological concept). In modern terminology one may interpret the experiences as long term memory consisting of knowlegde structures (in cognition), influencing the "impulses" which are the neurophysiological basis of short term memory (in perception).
To summarize: Bartlett is extending the neurophysiological concept of schemata to higher level mental processes, in my terms called cognition. He is putting emphasis on the active part of past experience on new experience. His concept of schemata is therefore a more psychological concept despite his idea to think in neurological terms about remembering.

6. Schemata and Cognitive Psychology

Ulric Neisser (1967, pp. 279) strenghtens the view that schemata are important in cognition, i. e. memory and thought. Following Bartlett he (Neisser 1967, 287) emphasizes that cogniton is active, or, as he said, constructive. Remembering and thinking are viewed as two-stage processes (Neisser 1967, 279). At the stage of the primary processes there is a highly parallel activity, and the secondary

process is sequential. Bartlett (1932) , as I mentioned , was interested to use the schema concept as a neurological concept. Ulric Neisser (1967, p. 281) is not further interested in the physiological realization of schemata: "Psychology deals with the organization and use of information, not with its organic representation in organic tissue." Two further points should be emphazised concerning Ulric Neisser's thinking on schemata. First, what is stored in memory are not phenomena or appearences but functions or acts. Neisser (1967, p. 279 & pp. 285): *"Stored information consists of traces of earlier acts, organized in way that correspond to the structure of these acts. ...The present proposal is, ..., that we store traces of earlier cognitive acts, not the products of those acts."* These organized systems storing this information he termed "cognitive structures" or "schemata". They play a particularly interesting role in learning and remembering. In thinking about remembering and reasoning the old cognitive psychology had to introduce terms like "searches through memory", "strategy", "turning around" to explicate processes not directly connected with shortterm-memory like perception and was forced to think of something or somone - a little man in the head, a homunculus - which is carrying out search in memory. In a formulation from Bartlett (1932, p. 206): *"An organism has somehow to acquire the ability to turn around upon its 'schemata' and to construct them afresh."*
For Ulric Neisser the problem of executive, as he termed it, is to explicate search in memory without referring to an agent - a homunculus - which would lead to an infinite regress. With the concept of a stored execute routine borrowed from computer-science, it is possible to give a solution to this problem. This is the second point to mention: the introduction of the idea of a program to cognitive psychology which originated in the work on TOTE-units of Miller, Galanter and Pribram (1960; see Seifert 1990 for a detailed discussion). Although he is thinking of the idea of a program from computer science as a useful concept for psycholgy, he is sceptic about "computer models" of psychological processes in general.
Neisser (1967, p. 296) mentions two strongly connected problems for computer models to become adequate models of perception and cognition and from which AI - symbolism and connectionsm - still suffers. Until today the program has to be established by a programmer (for recent critic see Reeke & Edelman 1988), and is not possible for a program to *"make major developmental changes in its own executive routine."* The solution of these problems presupposes the adequate description (computer languages for) of processes - in AI metaphorically known as debugging theory (Minsky 1980, pp. 22; 1984;) - and progress in the field of induction/learning (Putnam 1988; Holland et al. 1986).
In summary: It is important to see that for Neisser (1967, p. 302 & p. 303) cognitive psychology has to deal with acts or functions (in the tradition of Brentano, Husserl and Stumpf) and not with phenomena. The procedural character of acts is reflected by the distinction of implicit procedural knowledge and explicit declarative knowledge in cognitive psychology of music (Carterette & Kendall 1989, p. 147), originating from the declarative-procedural controvery in artificial intelligence. Schemata are memory structures - especially for long term memory - of stored mental acts and overt reactions, and the idea of search processes through "memory" is strongly connected with the idea of a stored computer-program, especially the idea of a subroutine. His concept of schema is purely psychological and especially to explicate cognition and not perception. Later in cognitive psychology - influenced by concepts from AI - the schema-idea dege-

nerated to a monolith in the explication of associative long term memory and as a building block in cognition (Iran-Nejad & Homaifar 1991; Rumelhart 1980; Rumelhart et al. 1988).

7. Inventing the Wheel: Schemata as Knowledge Structures in Artificial Intelligence

The idea to use representational structures like grammars, ATN's, semantic nets, production systems, frames, scripts etc. as formal description of internal representations (internal models of the environment or a cognitive domain) of cognitive systems for modeling perception, thinking, memory, and learning is central to cognitive science (for a review of many of these representational formalism for knowledge representation see Avron & Feigenbaum 1981; used in models of cognition Cohen & Feigenbaum 1982; as structures for memory Rumelhart & Norman 1988; Anderson 1988 for a general discussion). The can think of all these formalisms as a specification of the schema concept viewed as internal program. The main point to recognize is that we regard schemata as internal representations of the mind/brain, in particular at the "psychophysical level" (Bischoff 1966) of an organism on one side, i. e. as models of the environment or a cognitive domain *within* the mind/brain, and as conceptual descriptions of these internal schemata by a scientist - therefore as models *of* the mind - and in this case often termed body schema, world schema, motor schema, perceptual schema (Bischoff 1966; Arbib 1989, p. 9) on the other side. In this case it is appropriate in general to speak of schema-theory (Arbib 1989, p. 9).

In 1975 M. Minsky proposed *frames* as a general structure for scene analysis in vision and language understanding. He (Minsky 1980, 1) is speaking of a data-structure representing a stereotyped situation. The reaction to this idea from AI by different researchers was summarized by M. A. Arbib (1977, pp. 33) in the following way: "*(a) The "what a revelation" reaction of neophytes who had never before realized the importance of an internal representation of the world. ... (b) The "we've seen it all before" reaction ... (c) The "mature acceptance" reaction of workers in AI who have felt the need for a more general framework for the discussion of internal representations, and feel that recasting their work in the language of frames is a reasonable price to pay in moving toward such a generality.*"

One is finding all these reactions in the AI community and reaction (a) particularly in the cognitive psychology of music. AI-researchers who prefer their own formalisms and find the frame-concept too unformal can be subsumed under reaction (b).

Minsky's main idea was to establish a general more formally orientated framework for describing internal presentations, but after a carefull analysis of the frame-concept J. P. Hayes (1980, p. 58) summarizes the merits of the frame-movement: "*I believe that an emphasis on the analysis of ... processes of reflexive reasoning is one of the few positive suggestions which the 'frames' movement has produced. Apart from this, there are no new insights to be had there: no new processes of reasoning, no advance in expressive power.*" From cognitive psychology and artificial intelligence frames and scripts were critized as too inflexible in coping with new situations (Neisser 1979, pp. 52; Holland et al. 1986, pp. 12).

To summarize: One has to distinguish between schemata as internal representa-

tions *within* the mind/brain and their description by researchers as models of functioning *of* the mind/brain. These models *of* the mind may be termed in general schema-theory. To give these models an explicit description it is necessary to think of schemata as programs consisting of data structures and algorithms. Marvin Minskys goal was to establish a general formal framework for the description of internal representation (schemata) through frames. But Hayes' careful examination of the frame-concept showed that, in general, until now there is no preferable formalism. There are no candidates coping with the problems in building psychological adequate models of perception and cognition as formulated by Ulric Neisser (1967) and others (Gregory 1974; Holland et al. 1986; Reeke & Edelman 1988; Minsky 1980) so that AI has to solve these problems. In artificial intelligence and the modeling of psychological processes in cognitive psychology influenced by artificial intelligence the schema idea has lost its connection to neurophysiological thinking.

8. Piaget's Contribution to Cognitive Science

For the importance of Piaget's perspective for the schema idea in cognitive science in general and music modelling in particular, I will focus on his relevant ideas of *accomodation* and *assimilation* (Piaget 1981, pp. 32). In general cognition (i. e. perceiving, learning, thinking) is viewed by Piaget as an adaptive process. Adaptation consists of assimilation and accomodation. What is relevant here is to see that the idea of assimilation, i. e. the integration of a new entity in an existing structure or process without modifing the existing entity, has some formal counterpart in the knowledge structures developed in AI , especially the frame and script concept. Here also enters the now familiar idea of cooperative and competetive computation developed in AI and brain theory.
But up to now the idea of accomodation, i. e. the modification of the integrating entity, has no counterpart in AI and therefore is lacking. Frames and scripts were soon recognized as to be too inflexible (Neisser 1979, pp. 52 & p. 66; Holland et al. 1986, pp. 12). Perhaps schemata could be viewed as an elaboration of some AAM (angeborener Auslösemechanismus; innate releasing mechanism) as proposed by Konrad Lorenz in putting Kantian philosophy into an evolutionary biological perspective (The relation of Piaget's work to Konrad Lorenz' is discussed critically in Bischoff 1979; Arbib 1989, p. 410).
Piaget distinguishes the terms schema (*schéma*) and scheme (*schème*) (Arbib 1989, p. 410). In AI terminology a *schema* may be viewed as a frame (datastructure). Central to a scheme (a plan) is the idea of operability and it may therefore be regarded as a subroutine or algorithm. In the first english translations there was no distinction between these two concepts. To summarize: Piaget's work shows the problem to solve for AI if it will fruitfully contribute to a theory of cognition and perception. The main problem is to develop descriptions for structures which exhibit accomodation (Neisser 1979, pp. 49) - in AI-metaphor: a debugging theory. Further, computational models of adaptation have to incorporate cooperative and competitive computation.

9. "Neurologizing" Schemata Again

Donald O. Hebb (1949, pp. 60. & pp. 79; 1975, pp. 84 & pp. 104) proposed a theory of cell assemblies and phase sequences as mediating between neurophysiological structures or processes and psychological concepts like imagery, perceiving, thinking etc.

Cell assemblies are groups of neurons formed by learning and serving as embodiment of psychological function. Cell assemblies can be built up in a hierarchical fashion incorporating cell assemblies. Out of these are built phase sequences. As R. L. Gregory (1974) pointed out, the idea of phase sequences consisting out of cell assemblies can be viewed as a physiological correlate of Bartlett's (1932) schemata. Was Head only talking about schemata as an unspecified neurophysiological concept, one could regard Hebb's phase sequences as a specification of the schemata with the basic unit consisting of cell assemblies. In Hebb's theory one has to distinguish three main ideas:
a) the idea of neural reverberating circuits consisting of cell assemblies
b) which are randomly connected
c) following his well known neurophysiological postulate of synaptic learning through faciliation now at the basis of connectionnist modeling (see Braitenberg 1988).
The idea of reverberating circuits as the neurophysiological basis of short term memory was first spelled out by Lorente de Nó and is now well accepted in neuroscience. But wether there is a synaptic change as basis for long term memory is a difficult question and still not answered (see Schwartz 1988 and Palm 1988 for different points of view from neuroscience). Empirical research during the last thirty years on anatomical structure and physiological processes in the cortex cerebri indicate that it is doubtful wether *randomly* connected neural nets can serve as physiological embodiment of higher brain functions such as perceiving music, understanding language and thinking (Bullock 1961; Braitenberg 1988).
At the basis of these processes there must be *structured degenerate* repertoires of groups of neurons and *not randomly* connected nets (Edelman 1978, p. 60). This idea of highly structured cell assemblies as general organization principle of the neocortex is captured in the module or column concept of neural organization (Szentágothai 1977) Michael A. Arbib (1989) is connecting the idea of schemata with the well accepted module concept of neuroscience (for a carefull discussion of the module-concept in neuroscience and its implications for computational neuromusicology see Seifert 1991). These highly connected modules (the Hebbian cell assemblies and phase sequences) are the embodiments of schemata.

10. Modern Schema-Theory as Unifying Concept in Cognitive Science

I showed how the idea of schemata via cognitive psychology and artificial lost its connection with neurophysiological thinking. Since the 1980 there is a revival of "neurologizing" central concepts in cognitive psychology and artificial intelligence, e. g. in connectionism. This chapter will show how to relate research in symbolic artificial intelligence, neurophysiology, and cognitive psychology in the realm of schema theory. M. A. Arbib (1989; Arbib & Hesse 1986, p. 13; Arbib & Hill & Conklin, 1987) proposed schema-theory as a unifying approach to models of internal representation in cognitive science: "*We see "schema theory" as being at the heart of cognitive science, though our own development of schema theory by no means offers a view that has gained the consensus of cognitive scientists. For us, a "schema" is a "unit of representation" of a person's world. Schema theory is an attempt at an information -processing theory of the mind. It is a materialist theory in that it seeks to show that all human mental phenomena reduce to (complex) patterns of schema activation and that schemas are instantiated as dynamic processes in the brain.*"

These dynamic processes are physiologically realized by competitive and cooperative acting modules - the Hebbian cell assemblies - of the brain, connected in an hierarchical and heterarchical fashion. Having identified the neurophysiological correlate of schemata, which are the building blocks of cognition and perception, building the conceptual interfaces between different horizontal levels in mental processes, it is necessary to show how to "neurologize" symbolic AI-architecture. Lacking space and time let me sketch just one possibility proposed by M. A. Arbib (1989, pp. 176 especially p. 180) in "neurologizing" black-board-architecture which is also used in developing a model of music perception in implementing the generative grammar of tonal music (Jones & Miller & Scarborough 1988; Jones & Scarborough & Miller 1990; see Seifert 1990, pp. 359). The black-board architecture (Nii 1986), incorporating the idea of parallel cooperative computation in a symbolic fashion, was used first in the HEARSAY speech understanding system. HEARSAY is based on the "hypothezise-and-test paradigm", a paradigm (Arbib 1989, p. 176) also known as analysis-by-synthesis in speech perception (Halle & Stevens 1964) and the prime candidate for a general paradigm of perception (Gregory 1974).
The blackboard architecture consists of a general hierarchical datastructure, the *blackboard* (the database), and modules of *knowledge sources* (the program modules), often realized as production rules which are interacting with the various levels of the blackboard. Normally this architecture is implemented on a serial computer with the classical Von-Neumann-architecture and therefore needs an explicit *scheduler*, which determines the knowledge source to be activated in the next step. At each moment there are only a few data at the "focus of attention" of the *blackboard-monitor* on which the scheduler is acting. In Arbib's (1989, p. 178) view a knowledge source corresponds to a "*collection of related perceptual schemas*". These schemas correspond on the other hand to the activation of a few neural modules - groups of groups of neurons, or, in Hebb's terminology, phase sequences - (Abbib 1989, 180). But there is one major problem with this approach (Arbib 1989, 180): In AI knowledge sources are programs separately stored in memory and possibly copied and activated several times in the central processing unit. But a brain region like a group of modules cannot be copied , and if we identify knowledge sources via schemata with groups of modules, the following question arises: *How does a brain region support multiple simultaneous activations of its function*? This is the *problem of schema instantiation.* Arbib (1989, p. 180) stated the main problem for psychological research using the idea of a program stored in the brain like research in cognitive science in the following way: "*Thus a key question (the answer is still unknown) is, how, given a schema stored in the brain, may multiple instances be established in the cortex, each appropriately processing its own inputs and updating its own parameters.*"
Summary of the basic ideas: The schema is the basic functional unit of psychological analysis (especially perceptual or conceptual analysis and motor processes) and corresponds to individual percepts, concepts or psychological acts. This functional units are realized by the activity of modules or columns, the basic physiological and morphological structure of the cortex cerebri. In symbolic AI one can think of collections of schemata as knowledge sources - especially production rules - in blackboard architecture.

11. Conclusion

The history of the schema concept in different sciences shows that there are two directions: a specification and a generalization of the concept.

There is a specification in two senses. The location of the physical realisation of schema at first unspecified used by Head (1926) in neurology is specified in the Hebbian phase sequences (Hebb 1949; 1975) consisting out of cell groups. Current research in neuroscience is giving a further specification by the modul- or column concept. These schemata are called internal schemata.

An explanation in Carnap's sense of the schema-concept is coming from computer science especially artificial intelligence. In general is possible to think of schemata as programs consisting of data structures and algorithms describing processes realized by brain modules. These formalisms may be called external schemata of "schema theory". One short example in "neurologizing" blackboard architecture was given. The choice of such an architecture to model perception - for music perception see (Jones, Miller & Scarborough 1988; Jones, Scarborough & Miller 1990) - is supported by the "hypothesize-and-test" paradigm which is the best general framework for psychological research on perception (Gregory 1974; Halle & Stevens 1964). Thinking of stored processes (mental acts) by the brain is supported by the psychology of act (Carl Stumpf 1906) and research in cognitive psychology (Ulric Neisser 1967). In this sense most of musical knowledge involved in music perception is implicit represented (Carterette & Kendall 1989). Current research in neuroscience is indicating that cognitive connectionist modeling of higher brain functions with randomly connected networks seems to be implausible (Braitenberg 1988; Bullock 1961; Iran-Nejad & Homaifar 1991; Szentágothai 1977) and is supporting symbolic modeling as research strategy. But in cognitive musicology one has to evaluate these different forms of modeling at first with respect to their music theoretical plausibility.

Regarding schemata as realized by neurophysiological circuits, and as the building blocks of perceptual and cognitive behavior may be viewed as a generalization of this concept. The schema-concept is building an "interface" between research in neuromusicology and psychomuscicology and is mediating between lower mental functions like perception and higher mental function such as reasoning. These aspects of the "physical" and the "psychical" are worked out and are best understood in terms of Donald O. Hebb's (1949; 1975) neuropsychological theory of mental functions. Some central problems rest to be solved by artificial intelligence, if it as part of "cognitive science in music" will have some value for further research on modeling music perception - and as I believe it will have. Artificial intelligence has to cope with these central problems:

a) to develop a language for the description of cooperative and competitive processes: the debugging problem (Minsky 1984)

b) to cope with learning/induction (Holland et al. 1986; Putnam 1988)

c) to determine at the conceptual level (Genesereth & Nilsson 1987) in connection with research in musiology and psychology the music theoretical, perceptual and cognitive relevant entities for formalization.

A caveat from M. A. Arbib (1989, 84) may end this review of the schema-concept in science: "*As in a one day tour of a great city, many landmarks remain unvisited, and those we have visited, much that is important remains untold. However just as the one-day tour may help structure one's appreciation of the city's many delights for later so should give [this review] the reader some reference points ...*"

References

J. R. Anderson, "Methodologies for studying human knowledge", **Behavioral and Brain Sciences**, Vol. 10, 1987, pp. 467-505.

M. A. Arbib, "Parallelism, slides, schemas, and frames", in W. E. Hartnet (Ed.), **Systems: approaches, theories, applications. Including the Proceedings of the Eigth George Hudson Symposium Held at Plattsburgh, New York, April 11-12, 1975**, Reidel, Dordrecht-Holland, 1977, 27-43.

M. A. Arbib, **The metaphorical brain 2. Neural networks and beyond**. Wiley, New York, 1989.

M. A. Arbib, and M. B. Hesse, **The Construction of reality**, Cambridge University Press, Cambridge, 1986.

M. A. Arbib, E. J. Conklin, and J. Hill, **From schema theory to language theory**, Oxford University Press, New York, 1987.

A. Barr, and E. A. Feigenbaum (Eds.), **The handbook of artificial intelligence. Vol. I. Chap. III Knowledge representation**, Pitman, London, 1981, pp. 141-222.

F. C. Bartlett, "Critical notice of Head's *Aphasia and kindred disorders of speech*", **The British Journal of Psychology**, General Section, Vol. 17, pp. 154-161.

F. C. Bartlett, **Remembering: a study in experimental, and social psychology**, Cambridge University Press, Cambridge, 1932.

J. J. Bharucha, "Kognitive Musikpsychologie", in H. Bruhn, R. Oerter and H. Roesing (Eds.), **Musikpsychologie. Ein Handbuch in Schluesselbegriffen**, Urban & Schwarzenberg, Muenchen, 1985, pp. 123-132.

N. Bischoff, "Erkenntnistheoretische Grundlagenrobleme der Wahrnehmungspsychologie", in K. Gottschalk, Ph. Lersch, F. Sander and H. Thomae (Eds.), **Handbuch der Psychologie in 12 Bänden. 1. Band**. W. Metzger (Ed.), **Allgemeine Psychologie. I. Der Aufbau des Erkennens. 1. Halbband: Wahrnehmung und Bewußtsein** Verlag für Psychologie, Göttingen, 1966, pp. 21-78.

N. Bischoff, "Remarques sur Lorenz et Piaget: comment les «hypothèses de travail» peuvent-elles être nécessaires?", in M. Piatelli-Palmarini (Ed.), **Theories du langage - Theories de l'apprentissage. Le debat entre Jean Piaget et Noam Chomsky**, du Seuil, Paris, 1979, pp. 343-352.

V. Braitenberg, "Two views of the cerebral cortex", in G. Palm, and A. Aertsen (Eds.), **Brain Theory. Proceedings of the First Trieste Meeting on Brain Theory , October 1-4, 1984**, Springer, Berlin, 1986, pp. 81-96.

H. Bruhn, **Harmonielehre als Grammatik der Musik. Propositionale Schemata in Musik und Sprache**, Muenchen, Verlagsunion Psychologie, 1988.

T. H. Bullock, "The problem of recognition in an analyzer made of neurons", in W. A. Rosenblith (Ed.), **Sensory Communication**. Contributions to the symposium on principles of sensory communication, July 19-August 1, 1959, Endicott House, MIT, MIT-Press & Wiley, New York & Cambridge, Mass., 1961, 717-724.

E. C. Carterette and R. A. Kendall, "Human music perception", in R. J. Dooling and S. H. Hulse (Eds.), **The comparative psychology of audition: Perceiving complex sounds**, Erlbaum, Hillsdale, NJ, 1989, pp. 131-172.

P.-P. Changeux, **Der neuronale Mensch. Wie die Seele funktioniert - die Entdeckungen der neuen Gehirnforschung**, Reinbek bei Hamburg, Rowohlt, 1984.

P. R. Cohen, and E. A. Feigenbaum (Eds.), **The handbook of artificial intelligence. Vol. III. Chap. XI, Models of Cognition**, Pitman, London, 1982, pp. 1-74.

E. T. Cohn, **Musical form and musical performance**, Norton, New York, 1968.

G. M. Edelman, "Group selection theory and reentrant signaling: Atheory of higher brain function", in G. M. Edelman and V. B. Mountcastle (Eds.), **The mindful brain. cortical organization and the group-selective theory of higher brain function**, MIT-press, Cambridge, Mass., 1978, pp. 51-100.

H. G. Frank, "Informationspsychologie", in H. G. Frank, **Kybernetische Grundlagen der Pädagogik. Eine Einführung in die Pädagogistik für Analytiker, Planer und Techniker des didaktischen Informationsumsatzes in der Industriegesellschaft. 2. völlig neubearbeitete und wesentlich erweiterte Auflage. Zweiter Band. Angewandte kybernetische Pädagogik und Ideologie**. Baden-Baden: Agis, 1969, pp. 61-144.

M. R. Genesereth & N. J. Nilsson, **Logical foundations of artificial intelligence**, Kaufmann, Los Altos, 1987.

R. L. Gregory, "Choosing a paradigm for perception", in E. C. Carterette and M. P. Friedman (Eds.), **Handbook of perception. Vol. I. Historical and philosophical roots of perception**, Academic Press, New York, 1974, pp. 255-283.

M. Halle and K. N. Stevens, "Speech recognition: A model and a program for research" in J. A. Fodor and J. J. Katz (Eds.), **The structure of language. Readings in the philosophy of language**, Prentice-Hall, Englewood-Cliffs, NJ, 1964, pp. 604-612.

P. J. Hayes, "The logic of frames", in D. Metzing (Ed.), **Frame conceptions and text understanding**, de Gruyter, Berlin, 1980, pp. 46-61.

H. Head, **Studies in Neurology**, Vol. II, Hodder & Stoughton, London, 1920.

H. Head, **Aphasia and kindred Disorders of speech**, Vol. I, Cambridge University Press, Cambridge, 1926.

H. Head, and G. Holmes, "Sensory disturbances from cerebral lesions", **Brain** Vol. 34, 1911-12, pp. 102-254.

D. O. Hebb, **The organization of behavior. A neuropsychological theory**, Wiley, New York, 1949.

D. O. Hebb, **Einfuehrung in die moderne Psychologie. Neu uebersetzt nach der dritten voellig ueberarbeiteten Auflage von Hermann Rademacker**, Weinheim, Beltz, 1975.

H. Hensel, "Allgemeine Sinnesphysiologie", in W. D. Keidel (Ed.), **Kurzgefaßtes Lehrbuch der Physiologie, 2. überarbeitete Auflage**, Thieme, Stuttgart, 1970, pp. 361-368).

J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard, **Induction: Processes of Inference, Learning, and Discovery**, MIT-Press, Cambridge, Mass., 1986.

A. Iran-Nejad, and A. Homaifar, "Assoziative und nicht-assoziative Theorien des verteilten Lernens und Erinnerns", in S. J. Schmidt (Ed.), **Gedaechtnis. Probleme der interdisziplinaeren Gedaechtnisforschung**, Suhrkamp, Frankfurt a. M., 1991, pp. 206-249.

J. A. Jones, B. A. Miller and D. Scarborough, "A rule-based expert system for music perception. **Behavior Research Methods & Computers** Vol. 20, No. 2, 1988, pp. 255-262.

J. A. Jones, D. Scarborough and B. A. Miller, "GTSIM: A computer simulation of music perception", **Preprints of the Acts of the Conference "Music and Information Technology" - MAI 90** - 3-6 Octobre, Marseille (France), 435-441.

W. D. Keidel, "Allgemeine Informationsphysiologie der zentralen Hoerbahn", in W. D. Keidel

(Ed.), **Physiologie des Gehoers. Akustische Informationsverarbeitung. Einfuehrung fuer Aerzte, Biologen, Psychologen und Nachrichtentechniker**, Thieme, Stuttgart, 1975, pp. 164-226.

E. J. Kessler, C. Hansen, and R. N. Shephard, "Tonal schemata in the perception of music in Bali and in the west", **Music Perception**, Vol. 2, No. 2, 1984, pp. 131-165.

D. Lohmar, "Kants Schemata als Anwendungen von Kategorien auf Anschauungen. Zum Begriff der Gleichartigkeit im Schematismuskapitel der *Kritik der reinen Vernunft*", **Zeitschrift fuer philosophische Forschung**, Vol. 45, No. 1, 1991, pp. 77-92.

I. Kant, **Kritik der reinen Vernunft (1. Auflage 1781)**, Kants Werke Band IV, de Gruyter, Berlin, 1968.

I. Kant, **Kritik der reinen Vernunft** (2. Auflage 1787), Kants Werke Band III, de Gruyter, Berlin, 1968.

K. Koffka, **Principles of Gestalt psychology**, Kegan Paul, London, 1935.

D. M. Mackay, "Mind talk and brain talk", in M. S. Gazzaniga (Ed.), **Handbook of Cognitive Neuroscience**, Plenum, New York, 1984, pp. 293-317.

G. A. Miller, E. Galanter & K. Pribram, **Plans and the structure of behavior**, Holt, New York, 1960.

M. L. Minsky, "A framework for representing knowledge", in D. Metzing (Ed.), **Frame conceptions and text understanding**, de Gruyter, Berlin, 1980, pp. 1-25.

M. L. Minsky, "Adaptive control: from feedback to debugging", in O. G. Selfridge, E. L. Rissland, and M. A. Arbib (Eds.), **Adaptive control of ill-defined systems**, (NATO conference series. II, Systems science; v. 16; "Proceedings of the NATO Advanced Research Institute on Adaptive Control of Ill-defined Systems, held June 21-26, 1981 in Moretonhampstead, Devon, England"), Plenum Press, New York, 1984, pp. 115-126.

C. Nauck-Boerner, "Strukturen des musikalischen Gedaechtnisses. Anmerkungen zu formalen Modellen der Repraesentation", **Musikpsychologie. Jahrbuch der Deutschen Gesellschaft fuer Musikpsychology**, Vol. 5, 1988, pp. 55-66.

U. Neisser, **Cognitive Psychology**, Appleton-Century-Crofts, New York, 1967.

U. Neisser, **Kognition und Wirklichkeit: Prinzipien und Implikationen der kognitiven Psychologie**, Klett-Cotta, Stuttgart, 1979.

H. P. Nii, "Blackboard systems: The black board model of problem solving and the evolution of blackboard architectures. Part one. Blackboard systems: Blackboard application systems, blackboard systems from knowledge engineering perspective. Part two. **AI Magazine**, Summer & August, 1986, pp. 38-53 & pp. 82-106.

G. Palm, "Modellvorstellungen auf der Basis neuronaler Netzwerke", in H. Mandl and H. Spada (Eds.), **Wissenspsychologie**, Psychologie Verlagsunion, Muenchen, 1988, pp. 488-502.

J. Piaget, **Biologie und Erkenntnis. Über die Beziehungen zwischen organischen Regulationen und kognitiven Prozessen**, Fischer, Frankfurt a. M., 1974.

J. Piaget, **Meine Theorie der geistigen Entwicklung**, Fischer, Frankfurt a. M., 1983.

J. Pressing, "Improvisation: methods and models", in J. A. Sloboda (Ed.), **Generative processes in music. The psychology of performance, improvisation, and composition**, Clarendon, Oxford, 1988, 129-178.

H. Putnam, "Much ado about not very much", in S. R. Graubard (Ed.), **The artificial intelligence debate. False starts, real foundations**, MIT-Press, Cambridge, Mass., 1988, pp. 269-281.

G. N. Reeke, Jr., and G. M. Edelman, "Real brains and artificial intelligence", in S. R. Graubard (Ed.), **The artificial intelligence debate. False starts, real foundations**, MIT-Press, Cambridge, Mass., 1988, pp. 143-173.

Y. Reenpää, **Allgemeine Sinnesphysiologie**, Klostermann, Frankfurt a. M., 1962.

D. E. Rumelhart, "Schemata: The building blocks of Cognition", in R. J. Spiro, B. C. Bruce and W. F. Brewer (Eds.), **Theoretical issues in reading comprehension. Perspectives from cognitive psychology, linguistics, artificial intelligence, and education**, Erlbaum, Hillsdale, NJ, 1980, pp. 33-58.

D. E. Rumelhart, and D. A. Norman, "Representation in memory", in R. C. Atkinson, R. J. Herrnstein, G. Lindzey, and R. D. Luce (Eds.), **Stevens' handbook of experimental psychology. Second edition. Vol. 2: Learning and cognition**, Wiley, New York, 1988, pp. 511-587.

D. E. Rumelhart, P. Smolensky, J. J. McClelland, and G. E. Hinton, "Schemata and sequential thought processes in PDP models", in J. L. McClelland, D. E. Rumelhart, and the PDP research group (Eds.), **Parallel distributed processing. Explorations in the microstructure of cognition. Volume 2: psychological and biological models**, The MIT Press, Cambridge, Mass., seventh printing 1988, pp. 7-57.

J. T. Schwartz, "The new connectionism: Developing relationships between neuroscience and artificial intelligence", in S. R. Graubard (Ed.), **The artificial intelligence debate. False starts, real foundations**, MIT-Press, Cambridge, Mass., 1988, pp. 123-142.

U. Seifert, **Systematische Musiktheorie und Kognitionswissenschaft. Ein Beitrag zur Fundierung der Kognitiven Musikwissenschaft.** Ph. D. thesis, Universtität Hamburg, 1990.

U. Seifert, "Steps towards computational neuromusicology", **unpublished papers presented at the institute of musicology, university of Hamburg**, 19. 06. and 25. 06. 1991.

T. H Stoffer, "Modelle der kognitiven Verarbeitung und Repraesentation musikalischer Strukturen", in O. Neumann (Ed.), **Perspektiven der Kognitionspsychologie**, Springer, Berlin, 1985, pp. 147-184.

C. Stumpf, "Erscheinungen und psychische Funktionen", **Abhandlungen der koeniglich preussischen Akademie der Wissenschaften**, IV, Berlin, 1906, pp. 1-40.

J. Szentágothai, "The Ferrier Lecture , 1977. The neuron network of the cerebral cortex: a functional interpretation. **Proceedings of the Royal Society London**, Vol. 201, Series B, 1978, pp. 219-248.

E. Terhardt, "Methodische Grundlagen der Musiktheorie", **Musicologica Austriaca** Vol. 6, 1986, pp. 107-126.

PERCEPTUAL DIMENSIONS AND ACOUSTIC CORRELATES OF LAUGHTER

Silvia Girardi* and Graziano Tisato**

*Civica Scuola d'Arte Drammatica "P. Grassi", Milano
**Centro di Calcolo di Ateneo - Centro di Sonologia Computazionale
Università di Padova -Via S. Francesco 11, I-35121 Padova, Italy
E-mail: music01@ipdunivx.it Tel: +39-49-8283755 Fax: +39-49-8283733

Keywords: Perceptual Effects of Sound, Psychoacoustics, Acoustical Correlates of Emotions, Non-linguistic Communication, Parametric Representation of Vocal Sounds.

ABSTRACT

This paper is concerned with a methodological approach to the analysis of one of the most representative examples of non- linguistic expressions: laughter. It seems to us interesting to investigate the form of information related to perceived feeling and the way this information is transmitted through the auditory channel. The communicative content of laughter is correlated to some dynamic expressive forms of acoustic parameters, responsible for the pre-conceptual reaction of the brain, similar to that which happens in other non-verbal communications (Clynes, 1981), as well as in music (Fonagy, 1981). The identification of these forms having an innate emotional meaning allows an immediate application to music, both for synthesis and performance.

Very little is known about laughter: the scarce literature deals mainly with its physiological and psychological aspects and only rarely treats its acoustic parameters. This study tries to integrate the two aspects of the problem and looks for a correlation between the perceptual dimensions, derived from the factor analysis of appropriateness scores expressed by 20 listeners, and the acoustical parameters extracted by computer analysis.

The research steps are summarized as follows:

- a statistically suitable number of laughs were collected according to ten discrete types (joy, contempt, madness, etc.);
- the tokens were submitted to an auditory recognition test to obtain a choice between 16 labels;
- a factor analysis of appropriateness scores determined the perceptual latent dimensions (superiority/inferiority, responsiveness/inhibition, tension/relaxation, etc.) and factor scores;
- a structural model of the laugh was proposed to describe and compare the different tokens;
- 23 acoustic parameters (pitch variations, amplitude variations, noise-vowel alternations, formants position, etc.) were extracted by LPC and FFT analysis;
- the factor scores and acoustic parameters were correlated to determine which of them were significantly linked to the perceptual dimensions;
- a formant and LPC synthesis was made, order to verify the actual importance of the parameters;
- sound examples of possible musical application were synthesized using some of the dynamic expressive patterns identified .

INTRODUZIONE

Nell'ambito artistico e musicale si è sviluppato un settore interdisciplinare che ha coinvolto da una lato la ricerca teorica e scientifica (semiotica, linguistica, neurofisiologia, psicofisica, ecc.), e dall'altro la ricerca più propriamente applicativa e musicale, con lo scopo di affrontare fra l'altro le problematiche tipiche della comunicazione umana. Al di là del parallelismo ovvio, che si è potuto stabilire fra linguaggio parlato e linguaggio musicale (in cui si può individuare un lessico, una grammatica e, forse anche, una semantica), un particolare e meno teorico interesse riveste per la musica lo studio dell' espressione dei sentimenti e degli stati emozionali, e la formulazione delle regole con cui questo livello di comunicazione interferisce con gli altri livelli strutturali. Così come si dimostra l'esistenza di strutture sonore, sintattiche, semantiche, elaborate nei processi di produzione e ricoscimento del parlato, così esiste un livello strutturale "emozionale", che è espressione, a volte completamente involontaria, della paura, rabbia, gioia, ecc. Accanto e parallelamente, dunque, all'estrazione del significato linguistico, avviene un riconoscimento dello stato emotivo. Si può anzi sostenere che produzione e riconoscimento dell'emozione si situano ad un livello biologico preconcettuale molto più elementare, ma non per questo meno potente di quello linguistico. Ad esempio, posso non afferrare il senso del discorso che il mio interlocutore mi sta facendo, ma capire perfettamente che è in preda al terrore, all'ansia, ecc., e subire mio malgrado le medesime sensazioni. Questa trasmissione di informazione avviene anche se non sono in contatto visivo con la persona. Le tensioni mentali e muscolari di chi sta parlando si traducono in una variazione dinamica dei parametri acustici che è rilevata dall'ascoltatore e associata alla corrispondente emozione.

L'individuazione della forma assunta da queste variazioni dinamiche è stata, non a caso, l'oggetto di ricerca di molti studiosi (Langer, 1973, Fonagy, 1981, Sundberg, 1981) in campo musicale, visto che esse continuano a portare lo stesso significato emotivo anche in questo contesto. Le ricerche in questo settore si sono concentrate più sull'andamento dei parametri sovrasegmentali (intensità, altezza, durata), facilmente rilevabili e trasponibili in termini musicali, che sull'individuazione delle variazioni spettrali, comunque importanti nel rivelare le tensioni emotive, ma di utilizzo più problematico in una situazione musicale.

Nel campo della comunicazione non-verbale, Manfred Clynes ha condotto ricerche sperimentali nell'arco di un ventennio, tese a dimostrare l'esistenza di quelle che egli definisce "sentic forms", cioè forme espressive dinamiche, che hanno per noi un significato emotivo innato, e che si possono derivare, ad esempio, dalla tensione muscolare espressa attraverso la pressione di un dito (Clynes, 1981).

Le implicazioni, sia che si tratti di parametri sonori direttamente utilizzabili ai fini musicali, sia che (come in Clynes) siano tradotti con una opportuna trasformazione nell'ambito sonoro voluto, sono evidenti tanto nell'interpretazione, quanto nella scrittura musicale. La differenza con il passato, in cui si incontrano innumerevoli esempi di utilizzo di espressioni musicali che riproducono in maniera più o meno diretta gesti, emozioni, stati d'animo, ecc., consiste in una formalizzazione di questi processi più rigorosa ed analitica e meno soggetta all'intuizione e all'esperienza personale.

Le espressioni sonore (come il sospiro, il riso, il lamento, ecc.), che sono prive di un significato linguistico, sono estremamente interessanti, poichè la carica emotiva che riescono a trasmettere non è affidata al significato delle parole o mescolata ai fattori prosodici, ma si identifica con il pu-

ro gesto sonoro, così come avviene nella musica. I segnali non verbali, possono essere intesi tanto come segno quanto come comunicazione, definendo i segni semplicemente come risposte comportamentali o fisiologiche, distinti dalla comunicazione in quanto non richiedono la consapevolezza che esista un destinatario e che condivida un codice comune.
Non tutte queste espressioni, hanno un significato universalmente accettato, poichè, accanto ai fattori evolutivi della specie, agiscono componenti culturali differenti. Per questa ragione non è possibile generalizzare risultati ottenuti in una certa area culturale e anche linguistica (per il problema delle definizioni verbali a cui si va incontro in queste ricerche), senza confrontarli con quelli ricavati in altre aree.

LA RISATA

Il presente lavoro intende affrontare lo studio della risata come uno degli esempi di comunicazione non verbale più significativi dal punto di vista musicale, visto l'ampio spettro di situazioni espressive che può convogliare. Ne prenderemo, dunque, in considerazione solo il livello di informazione sonora, trascurandone il livello visivo, costituito, ad esempio, dalle espressioni facciali.
La ricerca si propone di integrare l'aspetto psicologico e quello acustico della risata, tentando una descrizione del fenomeno con gli opportuni parametri fisici, e una loro correlazione con le dimensioni percettive, dedotte dalla analisi fattoriale dei giudizi espressi in fase di ascolto.
Il tentativo di caratterizzare il fenomeno da questo punto di vista è totalmente nuovo per un argomento tradizionalmente oggetto di speculazione in altre discipline. Infatti le scarse pubblicazioni esistenti in letteratura affrontano prevalentemente aspetti fisiologici o psicologici, considerandone solo in due casi e parzialmente (Kori, 1985, Mowrer et al., 1987) la correlazione con i parametri acustici. Non risultano ricerche effettuate con soggetti appartenenti all'area linguistico-culturale italiana. Lo studio di Kori, a cui ci siamo ispirati, analizza unicamente le risate da lui prodotte, mentre in questo lavoro si analizzano esecutori diversi con diverse caratteristiche vocali.
Da molti studiosi della materia, il riso è visto come un segnale innato con una complessa storia evolutiva (Spencer, 1860, Darwin, 1890, Bergson, 1900, Van Hoof, 1972, Ceccarelli, 1988). L'etologo Eibl- Eibesfeldt ha studiato il fenomeno del riso e del pianto nei bambini piccoli e ne ha notato la comparsa persino nei bambini ciechi e sordi, che non potevano averli imitati (Eibl-Eibesfeldt, 1973). Argyle ha preso in considerazione le origini e lo sviluppo evolutivo dei segnali umani in una specie come i primati molto vicina all'uomo (Argyle, 1978).
Askenazy suppone che le contrazioni ritmiche di tutti i muscoli dello scheletro coinvolti nella risata abbiano origine in un ipotetico peacemaker di regolazione di questo complesso ciclo periodico, localizzato nel cervello posteriore (secondo esperimenti parziali, nel tronco cerebrale reticolare del cervello medio), e responsabile dell'individuale stile di risata (Askenazy, 1987).
Per quanto riguarda il meccanismo respiratorio nella risata, sono state effettuate su soggetti misure oggettive mediante pneumografi da Lloyd (Lloyd, 1938), da Svebak (Svebak, 1975) e da Fry e Hader (Fry et al., 1977).

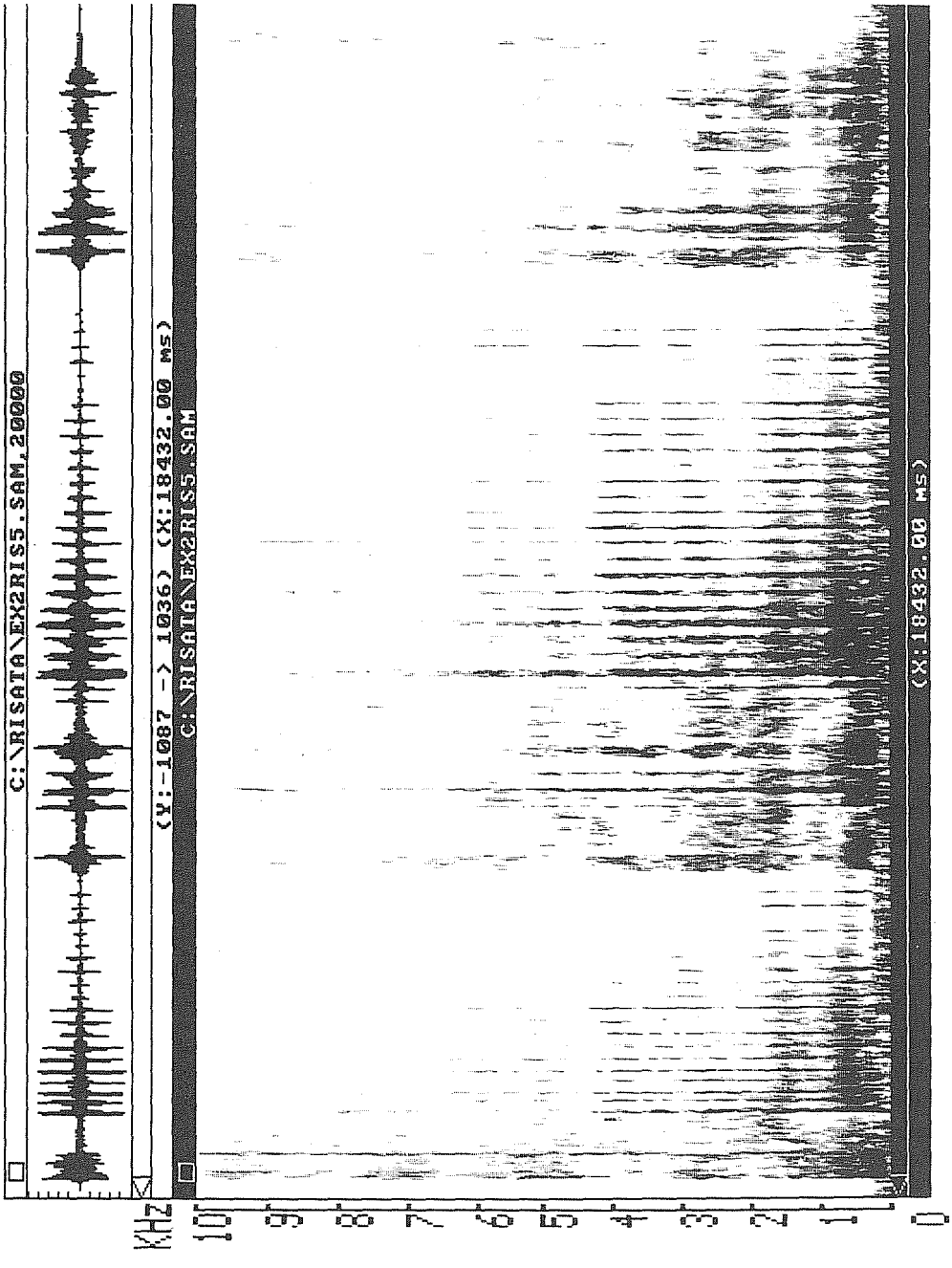


Fig. 1 Inviluppo di ampiezza e sonogramma della risata 5 dell'esecutore 2

SCHEMA STRUTTURALE E ANALISI DELLA RISATA

Sono state adottate le stesse metodologie di ricerca applicate nel campo della fonetica sperimentale (Williams et al.,1972, Kori, 1985), per l'estrazione dei parametri fisici, e quelle tipiche dell'analisi statistica multidimensionale, per l'individuazione delle componenti emotive principali (Scherer, 1982, Plutchik et al., 1980).

La prima fase del lavoro è consistita nella raccolta dei campioni sonori. Vista l'impossibilità di ottenere campioni sonori spontanei, cioè nel momento e nel luogo in cui si manifestano, l'esecuzione delle risate è stata richiesta a 10 soggetti educati, omogenei tra loro per area linguistico-culturale ed età, studenti della scuola di arte drammatica "P. Grassi" di Milano. La scelta dei soggetti è stata operata sulla base delle caratteristiche vocali personali quali il timbro e il rango della voce parlata, e della provenienza regionale, per formare un campione rappresentativo e valido statisticamente.

A ciascuno dei soggetti è stato innanzi tutto chiesto di leggere un brano (di circa 2') di Galileo Galilei il più naturalmente possibile. Dall'intera lettura sono stati estratti un campione di 10" di parlato abituale da cui ricavare i parametri di riferimento medio per le risate. E' stato quindi richiesto di produrre dieci differenti risate, in risposta alle dieci sottoelencate situazioni (per un totale di 10 risate x 10 esecutori = 100 campioni di risata):

1 - **Riso di gioia o sorpresa** per un avvenimento inatteso positivo.

2 - **Riso folle o isterico** in condizioni che non prevedono una simile reazione.

3 - **Riso di disprezzo o sfida** implicante superiorità sia intellettuale che sociale.

4 - **Riso falso, di convenienza** in una situazione di non- comprensione o non-sincerità.

5 - **Riso di solidarietà, amicizia** come reazione ad un evento comico fra amici oppure alla visione di un film di C.Chaplin.

6 - **Riso di timidezza o vergogna** in una situazione di imbarazzo.

7 - **Riso grasso, volgare** di abbandono di ogni ritegno nel proprio comportamento.

8 - **Riso di rabbia** per reazione ad una situazione di impotenza che rivela una componente aggressiva.

9 - **Riso di seduzione** in una situazione in cui si vuole attrarre una persona di sesso opposto.

10 - **Riso trattenuto**, di complicità in una situazione in cui non ci si può esprimere liberamente (es. scuola, teatro..).

Dopo l'esecuzione, al soggetto era consentita la replica nel caso in cui egli stesso giudicasse insoddisfacente la sua prestazione. Questo giudizio costituisce una prima fase di filtraggio qualitativo degli stimoli sonori generati (un successivo filtro uditivo è rappresentato dal gruppo di ascolto, omogeneo a quello degli esecutori, che verifica l'appartenenza effettiva dello stimolo ascoltato ad un certo tipo di risata). Sono stati registrati separatamente su tracce sincrone il segnale verbale con un microfono, e il segnale glottico mediante un elettroglottografo, per permettere la rilevazione esterna dell'altezza del suono vocalizzato. L'esame di questo tracciato consentiva in fase di analisi il controllo dei casi di presenza dubbia di vocalizzazione.

La regolazione dei livelli degli strumenti e la distanza del microfono dalla bocca dell'esecutore sono state stabilite una volta per tutte con il primo soggetto. Questa taratura comporta il problema di prevedere a priori quale possa essere il livello massimo di intensità raggiunto, ma questo svantaggio è compensato dal fatto che lo stesso riferimento di intensità assoluta consente il con-

fronto delle misure acustiche di tutti i soggetti. Gli stimoli sonori, opportunamente convertiti, sono stati analizzati con il sistema ICMS (Interactive Computer Music System, G. Tisato, 1975-1991), al CSC dell'Università di Padova.
Sono stati finora analizzati 40 campioni di risata, di 4 dei 5 soggetti maschili: si pensa di analizzare in un secondo tempo le voci femminili, poichè è necessario riportare con un certo fattore di scala i parametri legati all'altezza del suono e alla posizione delle formanti, per poterli effettivamente confrontare con i dati estratti dalle voci maschili.
A questo punto si è presentato il problema di definire dei criteri per poter confrontare tra loro gli stimoli ottenuti, dal momento che risultavano assai diversi per quanto riguarda i parametri di durata, di ritmo (inteso come rapporto tra le durate degli elementi componenti le risate), di contenuto spettrale (fig. 1), di intensità e di altezza della frequenza fondamentale (fig. 2). Si è deciso di segmentare questi "oggetti-risata", innanzi tutto da un punto di vista fisico-acustico, in gruppi di respiro, considerando come un unico gruppo gli eventi sonori compresi fra due inspirazioni successive. La segmentazione adottata tiene conto del ritmo naturale imposto dalla necessità della respirazione, e risulta particolarmente evidente sia dal tracciato della forma d'onda che dal sonogramma (fig. 1). Con questo criterio, il rango della durata dei gruppi è portato a limiti temporali ragionevoli (0.3-9 secondi), consentendo un confronto effettivo fra le parti. Il numero dei gruppi di respiro individuati nelle risate varia da 1 fino a 15 nei gruppi di durata maggiore.
All'interno dei gruppi di respiro si procede ad una ulteriore suddivisione strutturale caratteristica della risata. I 4 elementi fondamentali, che vengono proposti in questo lavoro, quali descrittori del fenomeno, sono i seguenti (fig. 3-5):
Elemento A: elemento inspiratorio iniziale. E' costituito dalla ripresa del fiato ed è un elemento tipico della risata, non presente nel linguaggio abituale, poichè:
1 - è (a volte) perfettamente udibile, sia esso vocalizzato o non vocalizzato;
2 - la produzione del suono avviene in inspirazione forzata, con un meccanismo opposto a quello del parlato (almeno per l'italiano), dovuto alla particolare tensione muscolare volontaria o involontaria;
3 - a volte si presenta vocalizzato con altezza estremamente elevata (anche più di 2 ottave) rispetto al parlato.
Elemento B: elemento espiratorio iniziale. E' costituito dal primo suono emesso dopo l'inspirazione e può essere sia vocalizzato che rumoroso.
Elemento C: alternanza vocale-rumore. E' un altro elemento caratteristico della risata ed è costituito da una successione di suoni vocalici alternati a rumore, riconducibile alla fricativa aspirata /h/. Il rumore è generato dal passaggio del flusso d'aria turbolento attraverso il restringimento della glottide, e segue quasi sempre immediatamente senza soluzione di continuità il suono vocalizzato.
Elemento D: elemento ritmicamente irregolare. Costituisce un suono isolato oppure un insieme di suoni e/o rumori non classificabili con un'alternanza (fig. 3,5, gruppo 5, dopo l'elemento B, al ms 16270) e non situati in prima posizione dopo l'inspirazione (fig. 3,5, gruppo 3, dopo l'elemento B, al ms 7290).
All'interno del singolo gruppo gli elementi A e B sono unici, mentre gli elementi C e D possono ripetersi e possono scambiarsi la posizione, come illustrato in figura 4. Il primo gruppo di cia-

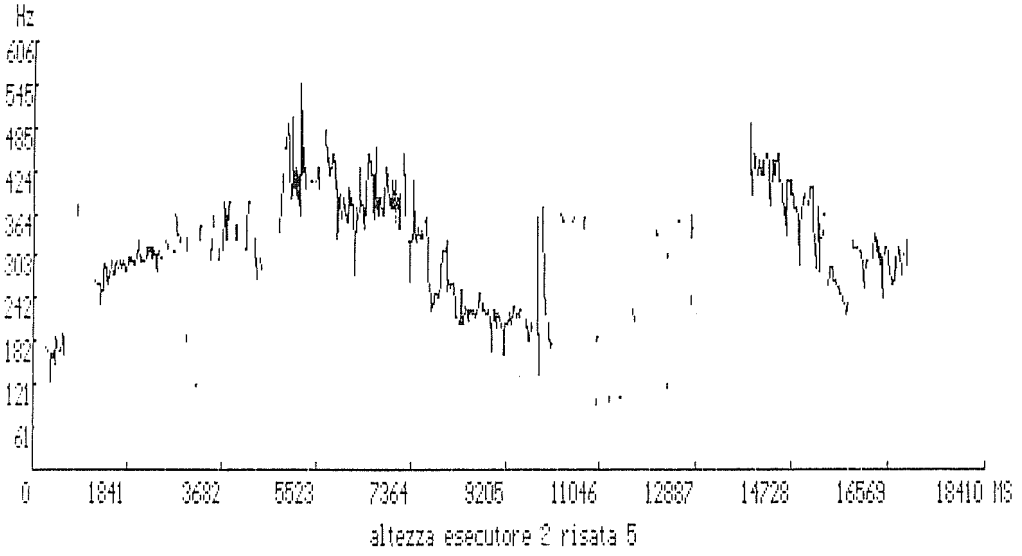


Fig. 2 Andamento dell'altezza nella risata 5 dell'esecutore 2

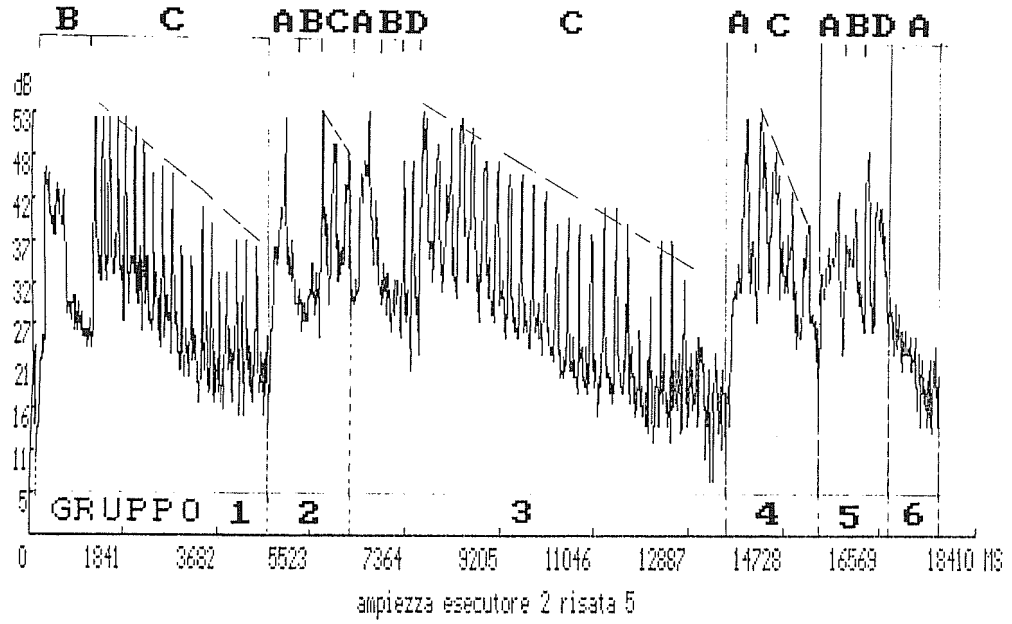


Fig. 3 Ampiezza e schema strutturale della risata 5 dell'esecutore 2

scuna risata non contiene mai l'elemento A e questo è uno dei fattori che sembrano distinguerla dal pianto (Lloyd, 1938).

Dal punto di vista fonetico, i suoni individuati nella risata e costituenti gli elementi sopraelencati sono: le vocali, compresi suoni vocalizzati propri di altre lingue o dialetti (si è deciso infatti di non limitare la libertà espressiva dei soggetti,in fase di esecuzione, all'uso di una sola vocale, come avveniva nello studio di Kori), consonanti sonore come le nasali /m/ o /n/, suoni fricativi e occlusivi (gutturali, velari, ecc.). I suoni esistenti sono stati individuati in base alla posizione delle prime 3 formanti sul sonogramma o dai dati di analisi LPC (Linear Predictive Coding), e quindi trascritti con la notazione dell'alfabeto fonetico internazionale in una specie di partitura di riferimento.

In fig. 3 è stata evidenziata la struttura della risata n. 5 dell'esecutore 2 (classificata come "gioia" e "serenità" dagli ascoltatori, in coincidenza con le intenzioni dell'autore) sul grafico dell'andamento della frequenza fondamentale e della ampiezza. Si osserva che il primo gruppo è composto dagli elementi B e C: secondo la regola prima enunciata nel primo gruppo non compare l'elemento A. E' anche evidente tanto dalla forma d'onda quanto dal sonogramma l'alternanza vocale-rumore che costituisce l'elemento strutturale C. Il valore trovato è in questo caso di 5,7 alternanze/secondo (visto che si contano 19 alternanze in un intervallo relativo all'elemento C di 3.31 s). Sono evidenziati poi gli altri gruppi (in totale 6). Si può notare come in questo esempio l'ampiezza diminuisca progressivamente nei vari elementi C, la durata dei segmenti vocalizzati cali parallelamente, mentre aumenta invece l'intervallo temporale fra i picchi vocalici. Definita dunque la struttura di base, si elencano i parametri acustici che sono stati estratti dall'analisi con il sistema ICMS. Fra parentesi si forniscono gli estremi di variazione dei parametri, e, se applicabili, l'eventuale valore medio e la deviazione standard:

1) DUR (ms) Durata del gruppo di respiro (220 - 10980 ms, 2473, 1810).

2) F0MAX (Hz) Altezza massima dei suoni vocalizzati nel gruppo (90 - 625 Hz, 238, 139).

3) F0MED (Hz) Altezza media dei picchi di frequenza delle vocali di tutto il gruppo (86 - 579 HZ, 199, 115).

4) F0VAR (Hz) Variazione dell'altezza media F0MED rispetto al parlato (essendo stati calcolati in precedenza la frequenza fondamentale media e l'intensità media del campione di parlato) (-57 - 435 Hz).

5) F0RAN (Hz) Rango di variazione dell'altezza: differenza fra massimo e minimo valore dei picchi di altezza dei suoni vocalici dell'elemento C (-252 - 398 Hz).

6) F0NOR (Hz/s) Rango di variazione dell'altezza normalizzato: si ottiene dividendo F0RAN per la durata dell'elemento C (-234-537).

7) F1MED (Hz) Posizione media in frequenza della prima formante nel gruppo (250 - 1100 Hz, 587, 269).

8) F2MED (Hz) Posizione media in frequenza della seconda formante nel gruppo (1000 - 2100 Hz, e quindi soprattutto vocali /a/, /e/ /ɛ/, 1333, 547).

9) F3MED (Hz) Posizione media in frequenza della terza formante nel gruppo (1800 - 3000 Hz, 2177, 916).

10) F0ESP (Hz) Altezza del suono di espirazione nell'elemento B, se effettivamente è vocalizzato (81 - 645 Hz, 135, 133).

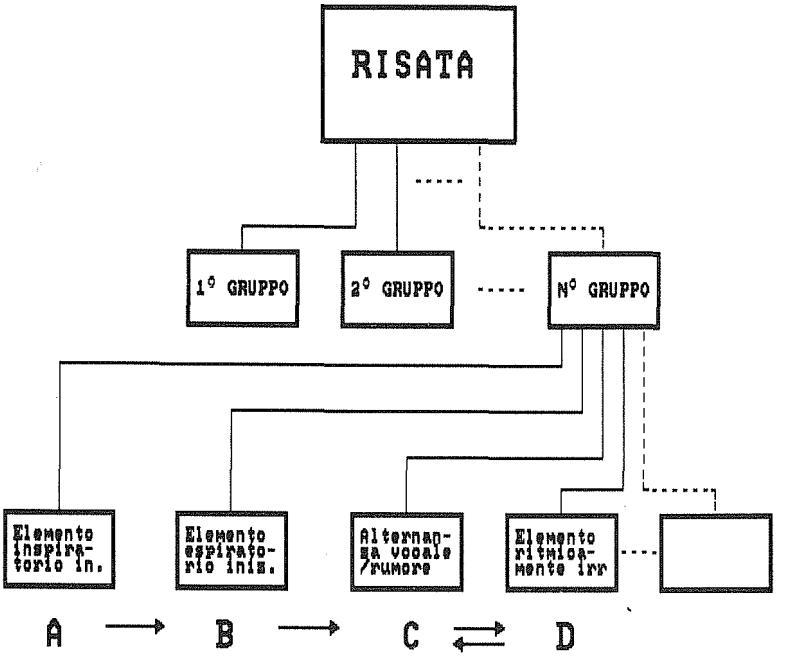


Fig. 4 Schema strutturale di una risata

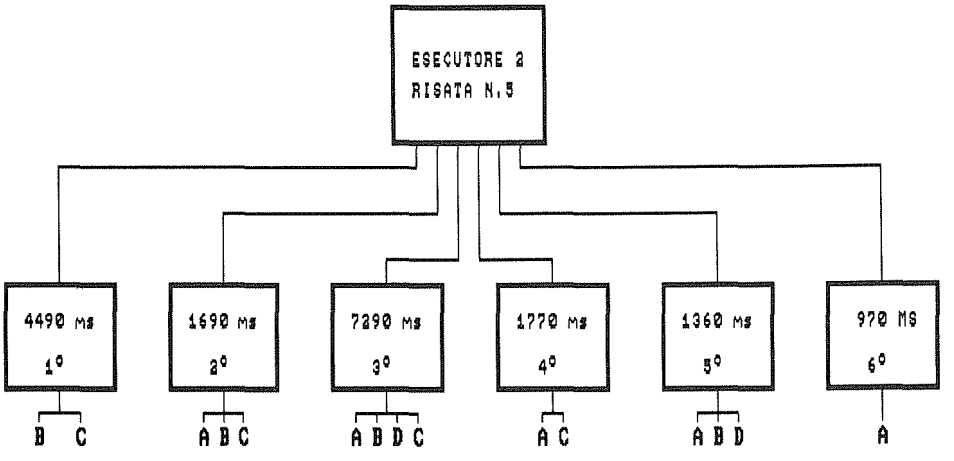


Fig. 5 Schema strutturale della risata 5 dell'esecutore 2

11) DURESP(ms) Durata dell'elemento B (50 - 2100 ms, 297, 326).
12) AMPESP (dB) Valore di intensità dell'elemento B (14 - 53 dB, 30, 19).
13) F0INS (Hz) Altezza del suono di inspirazione nell'elemento A, se effettivamente è vocalizzato (80 - 606 Hz, 176, 156).
14) DURINS(ms) Durata dell'elemento A (210 - 2440 ms, 485, 344).
15) AMPINS (dB) Valore di intensità dell'elemento A (15 - 54 dB, 33, 15).
16) NUALT (Hz) Numero di alternanze vocale-rumore al secondo dell'elemento C (2.21-10.3 Hz, 3.12, 2.69).
17) DURVO (ms) Durata.media delle vocali nelle alternanze (elemento C) (53 - 380 ms, 96, 86).
18) DURALT (ms) Durata media dell'intervallo fra picchi vocalici nelle alternanze (105 - 594 ms, 131, 117).
19) DURVA (ms) Variazione dell'intervallo temporale fra le vocali nelle alternanze (elemento C). Il parametro assume un valore positivo quando si ha un rallentamento progressivo nella velocità di emissione dei suoni della sequenza (-200 - 290 ms).
20) AMPRA (dB) Rango dei picchi di ampiezza delle vocali dell'elemento C (-28 - 24 dB).
21) AMPNO (dB/s) Rango di ampiezza delle vocali nelle alternanze normalizzato dividendo AMPRA per la durata di C (-14 - 44 dB/s).
22) ESPVO (dB) Variazione di ampiezza fra il suono di espirazione iniziale B e il suono vocalico seguente. Questo parametro fornisce una informazione di carattere intonativo, essendo positivo o negativo a seconda che l'accento cada sul primo suono, costituente l'elemento espiratorio iniziale, oppure sui seguenti (-29 - 28 db).
23) AMPVAR (dB) Variazione di ampiezza media rispetto al parlato (-30 - 19 dB).
Il rango di variazione dell'ampiezza (AMPRA, AMPNO) e dell'altezza (F0RAN, F0NO), possono avere anche valori negativi. Per convenzione si usa il segno negativo, quando i valori di ampiezza o frequenza crescono nel tempo, poichè questo è il caso meno frequente.
Tutte le operazioni relative ai valori di frequenza sono state effettuate su scala logaritmica per avvicinarsi al meccanismo percettivo dell'orecchio.
Il campione di risata nel suo complesso viene così rappresentato da una matrice in cui le colonne caratterizzano gli n gruppi di respiro (che sono risultati 315 in totale per le 40 risate) e le 23 righe rappresentano la variazione dei singoli parametri al variare nel tempo (vedi fig. 7 per i parametri della risata 1 dell'esecutore 2).

DIMENSIONI PERCETTIVE DELLA RISATA

Fino a questo punto sono stati considerati solo gli aspetti acustici del fenomeno. Le possibilità che il contenuto comunicativo della risata passi attraverso l'ascolto non è assodato in linea di principio e uno degli obbiettivi del lavoro era innanzi tutto quello di provare che gli indici acustici sono portatori di informazioni, e poi quello di individuare i parametri più significativi da questo punto di vista e trovare delle relazioni eventuali tra le loro variazioni e quelle della comunicazione trasmessa.
E' stato organizzato un gruppo di ascolto che esprimesse un giudizio sulla comunicazione ricevuta attraverso il canale uditivo. Il gruppo d'ascolto era costituito da venti persone di sesso maschi-

	gioia	incredulità	follia	superiorità	disprezzo	sorpresa	serenità	nevrastenia	imbarazzo	falsità	compiacenza	volgarità	rabbia	impotenza	civetteria	ilarità	soffocata	
n	15	5	1	1		9	6		1	2	1							< 1 >
n	2		14	3	4	1		8	1			3	3					< 2 >
n	3	2		9	5	1	2	1	2	1	6			1	2			< 3 >
n	2	2		9	1	4	3	1		2	6	2			2			< 4 >
n	11	2			3	4	10				3	1						< 5 >
n	1	1				1	3		12	3	4			2	11	3		< 6 >
n	9	3	1	1	2	4		2			3	12						< 7 >
n	2			1	2	1	1	4		5		1	7	13	3	2		< 8 >
n				7	3		4		1	5	9			1	5	2		< 9 >
n	4				2	1	3		3		1			2	2	19		<10 >

Fig. 6 Punteggi del test di adeguatezza per le 10 risate dell'esecutore 5 . Riquadrati sono i punteggi massimi ottenuti in coincidenza con l'intenzione dell'esecutore (sottolineata).

	Parametri Acustici	Gruppo 1	Gruppo 2	Gruppo 3
1	DUR	4290	5550	3720
2	F0MAX	400	400	392
3	F0MED	331	312	283
4	F0VAR	188	169	139
5	F0RAN	156	196	192
6	F0NOR	43	113	81
7	F1MED	1000	1100	600
8	F2MED	1500	1800	1500
9	F3MED	2800	2900	2400
10	F0ESP	0	400	323
11	DURESP	1130	1220	700
12	AMPESP	46	53	53
13	F0INS	0	250	400
14	DURINS	0	570	520
15	AMPINS	0	53	49
16	NUALT	3.87	4.65	4.68
17	DURVO	174	114	123
18	DURALT	250	204	214
19	DURVA	-30	-20	10
20	AMPRA	5	19	23
21	AMPNO	1.38	11.04	9.78
22	ESPVO	6	2	1
23	AMPVAR	8.57	6.47	3.72

Fig. 7 Parametri acustici dei 3 gruppi di respiro formanti la risata 1 dell'esecutore 2

le e femminile, omogenee tra loro e con gli esecutori, rispetto all' età e al livello culturale. L'esperimento di ascolto è stato binaurale e in campo diffuso per ricreare condizioni percettive abituali.
Gli ascoltatori hanno classificato gli stimoli sonori, che sono stati loro presentati in un ordine casuale, collocandoli all' interno di 16 insiemi individuati da etichette verbali (**gioia, incredulità, follia, superiorità, disprezzo, sorpresa, serenità, nevrastenia, imbarazzo, falsità, compiacenza, volgarità, rabbia, impotenza, civetteria, ilarità soffocata**). Le etichette verbali sono state fissate per poter avere un criterio di valutazione del significato, attribuito dagli ascoltatori, con un campo di variabilità minimo, rispetto ad una libera definizione. La scelta delle etichette ha anche tenuto conto delle definizioni date dagli esecutori stessi.
Per mettere in relazione tra loro i giudizi di adeguatezza forniti dal gruppo d'ascolto è stato utilizzato un programma di analisi statistica multivariata (Factor della SPSS e SAS), che permette di individuare gli eventuali assi di riferimento di uno spazio semantico, lungo i quali si ordinano i dati, altrimenti difficimente decifrabili. I giudizi espressi in sede di test sono stati raccolti e convertiti in un punteggio che può variare tra 0 e 20, rappresentando semplicemente il numero di persone, che hanno scelto di associare quella etichetta verbale alla risata ascoltata, esprimendo in questo modo un giudizio di adeguatezza. Il punteggio massimo raggiunto si è registrato per il campione 10 dell' esecutore 5 con $n = 19$. Anche nei casi meno favorevoli il punteggio massimo non è mai sceso sotto $n = 7$.
Nella maggior parte dei casi il massimo punteggio corrisponde all' etichetta che esprime l'intenzione dell'esecutore (vedi esempio in fig. 6). Questo dimostra che l'identificazione è quasi sempre possibile in base ai parametri acustici.
L' analisi multidimensionale ha estratto 5 fattori, i cui assi sono stati opportunamente ruotati mediante la procedura varimax (fig. 8). Il più forte di questi fattori, costituente una dimensione che definiremo per comodità "spontaneità-falsità", è nettamente bipolare e presenta ai suoi estremi le etichette relative alla "gioia" e "sorpresa" da una parte e "falsità", "compiacenza" e "civetteria" dall'altra. La seconda dimensione "tensione-rilassamento" oppone "follia" e "nevrastenia" all'etichetta "serenità". La terza dimensione "superiorità- inferiorità" vede da un lato "disprezzo" e "superiorità" e dall'altro "imbarazzo". La quarta dimensione "aggressività- sottomissione" ha da un lato "rabbia" e "impotenza" (mancano etichette fra quelle da noi proposte che si siano disposte sul semi-asse opposto). La quinta dimensione, infine, "reazione-inibizione", vede da un lato "incredulità" e "sorpresa" e dall'altro l'"ilarità soffocata". Come si vede quest'ultima dimensione ha una certa vicinanza con il primo fattore: la differenza consiste in una connotazione più emotiva del fattore 1, mentre il 5 fattore accentua una componente di risposta comportamentale.
Come secondo passo dell' analisi, calcolati i cosiddetti "factor scores" di ciascun campione per ciascuno dei cinque fattori, si è fatta la correlazione (con il metodo di Pearson) fra questi e i parametri acustici. In fig. 9 è riportata le matrice di correlazione ottenuta. Si osserva che le variabili acustiche riportano correlazioni significative con 4 dei fattori estratti.
Il fattore 1 ("spontaneità-falsità") è significativamente ($p = .001$) correlato con i seguenti parametri :
2) F0MAX picco di altezza dei suoni vocalizzati (-.3386), 3) F0MED Altezza media in tutto un gruppo (-.3124), 4) F0VAR variazione di F0MED rispetto al parlato (-.3512), 13) F0INS altezza

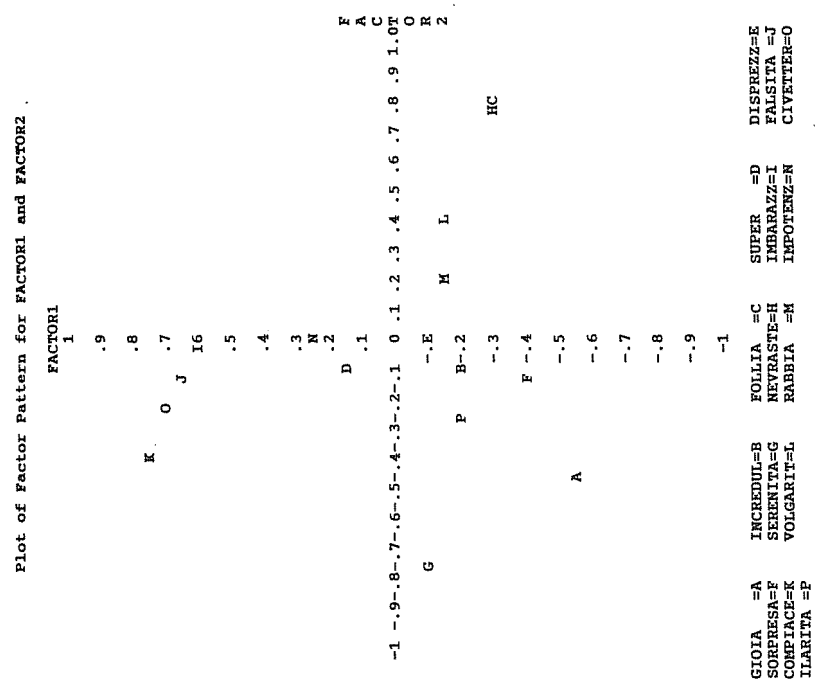


Fig. 8 Spazio dimensionale dei fattori 1 "spontaneità-falsità" e 2 "tensione-rilassamento".

Correlations:		FACTOR1	FACTOR2	FACTOR3	FACTOR4	FACTOR5
1	DUR	-.1296	-.1201	.0792	-.1465*	-.2631**
2	FOMAX	-.3386**	.2016**	-.2062**	.0006	.1418*
3	FOMED	-.3124**	.1825**	-.2508**	.0072	.1498*
4	FOVAR	-.3512**	.2299**	-.2795**	.0047	.1844**
5	FORAN	-.1608*	.0563	.0509	.0221	.0617
6	FONOR	-.0802	.0642	-.0727	.0551	.0142
7	F1MED	-.1255	.2037**	.2036**	.0453	.2580**
8	F2MED	.0419	.0563	.0773	.1171	.2677**
9	F3MED	-.0304	.0690	.1367*	.0497	.2410**
10	FOESP	-.1556*	.0717	-.1385*	.0169	.0226
11	DURESP	.0011	-.0498	.0699	.0449	-.0010
12	AMPESP	-.0068	-.0196	.0173	.0795	.1595*
13	FOINS	-.2709**	.1466*	.1846**	-.1147	.3280**
14	DURINS	.1360*	-.0406	.0167	-.0065	-.0945
15	AMPINS	-.1695*	.1633*	.1088	-.0462	.2184**
16	NUALT	-.0304	-.0247	.0385	-.0270	-.0261
17	DURVO	-.0195	.0057	.2014**	-.0579	-.0537
18	DURALT	-.0601	-.0375	.2077**	-.0614	-.0937
19	DURVA	-.0727	.0491	.1282	.0638	-.0462
20	AMPRA	-.0882	-.2654**	.0895	-.1185	.0181
21	AMPNO	.0440	-.1304	-.0704	.0090	.0555
22	ESPVO	-.1451*	.0208	-.1642*	.0239	-.1155
23	AMPVAR	-.2044**	.4053**	.1699*	-.0008	.4342**

N of cases: 315 Significance: * - .01 ** - .001

Fig. 9 Correlazioni fattori percettivi-parametri acustici di135 gruppi di respiro di 40 risate.

del suono di inspirazione (-.2709), 23) AMPVAR variazione di ampiezza rispetto alla media (-.2044). Con il fattore 1 sono correlati (con p = .01) anche i parametri seguenti: 5) F0RAN rango di variazione dell'altezza (-.1608), 10) F0ESP altezza del suono di espirazione (-.1556), 15) AMPINS ampiezza del suono di inspirazione (-.1695), 22) ESPVO variazione di ampiezza tra il suono espiratorio e quello della vocale seguente (-.1451).
La dimensione 2 "tensione-rilassamento" è significativamente (p = .001) correlata ai seguenti parametri: 2) F0MAX (.2016), 4) F0VAR Variazione di F0 media rispetto alla media del parlato (.2299), 7) F1MED Frequenza del primo formante (.2037), 19) DURVA Rango dell' ampiezza per le vocali (-.2654), 23) AMPVAR Variazione di ampiezza rispetto al parlato (.4053). Sono correlati al fattore 2 con significatività p = .01 i seguenti parametri: 3) F0MED (.1825), 13) F0INS Altezza del suono di inspirazione (.1466), 15) AMPINS Ampiezza del suono di inspirazione (.1633). Il fattore 3 "superiorità-inferiorità" è significativamente correlato ai seguenti parametri: 2) F0MAX altezza massima per i suoni vocalizzati (-.2062), 3) F0MED (-.2508), 4) F0VAR Variazione di F0MED rispetto al parlato (-.2795), 7) F1MED Frequenza della prima formante (.2036), 13) F0ins Altezza del suono di inspirazione (.1846), 17) DURVO Durata media delle vocali nelle alternanze (.2014), 18) DURALT Durata media dell'intervallo fra picchi vocalici (.2077), mentre con p = .01 si hanno : 22) ESPVO Var. di ampiezza tra suono espirato e vocale seguente (-.1642), 23) AMPVAR Var. di ampiezza rispetto al parlato (.1699).
Il fattore 5 "reazione-inibizione" è correlato con significatività p = .001 ai seguenti parametri: 1) DUR Durata dei gruppi di respiro (-.2631), con 7) F1MED (.2580), 8) F2MED (.2677), 9) F3MED (.2410), 15) AMPINS Ampiezza del suono di inspirazione (.2184), 23) AMPVAR Var. di ampiezza rispetto al parlato (.4342). Sono correlati con p = .01 i parametri: 3) F0MED (.1498), 4) F0VAR Var. di F0MED rispetto al parlato (.1844), 12) AMPESP Ampiezza del suono di espirazione in. (.1595). I parametri acustici 6) F0NOR, 11) DURESP, 14) DURINS, 16) NUALT, 19) DURVA, 21) AMPNO, non sono correlati significativamente con alcun fattore, facendo supporre che il loro contributo al riconoscimento di un particolare tipo di risata non sia determinante (ad esempio, NUALT la frequenza dell'alternanza vocale-rumore non influisce nell'individuare una particolare dimensione).

CONCLUSIONI

Le conclusioni che si possono tirare a questo stadio del lavoro sono estremamente parziali e riduttive, per quanto riguarda l'analisi dei campioni, visto che non ci si vuole limitare alle sole voci maschili e che si intende procedere anche con quelle femminili, e visto che resta da verificare con la sintesi l'importanza effettiva dei singoli parametri individuati e correlati. Dal punto di vista dell'applicazione musicale, d'altra parte, le regole minime estratte dalla correlazione delle dimensioni percettive con i parametri acustici possono trovare una interessante verifica in una situazione musicale, come si cercherà di mostrare con esempi didattici di sintesi.

RINGRAZIAMENTI

Gli autori desiderano esprimere la loro gratitudine a Shiro Kori dell'Università di Osaka e a Franco Ferrero del Centro di Studio per le Ricerche di Fonetica del CNR di Padova per il loro indispensabile aiuto.

REFERENCES

M. Argyle, **Il corpo e il suo linguaggio**, Zanichelli, Bologna, 1978.

J.J.M. Askenazy, "The functions and disfunctions of laughter", **Journal of General Psychology**, 114(4), 1987, pp. 317-334.

H. Bergson, **Le rire**, Alcan, Paris, 1900.

F. Ceccarelli, **Sorriso e Riso**, Einaudi, Torino, 1988.

M. Clynes and N. Nettheim, "The living quality of music: Neurobiologic basis of communicating feeling", in M. Clynes (Ed.) **Music, mind and brain**, Plenum Press, London, 1981, pp. 47-82.

C. Darwin, **Expression of the emotions in man and animal**, New York, Appleton Ed., 1890.

I. Eibl-Eibesfeldt, "The expressive behavior of the deaf and blind born", in M.V. Cranach and I. Vine (Eds.), **Social communication and movement**, Academic Press, New York, 1973.

I. Fonagy, "Emotions, voice and music", in **Research aspects on singing**, Royal Swedish Academy of music Ed., N. 33, Stockholm, 1981, pp. 51-79.

S. Girardi, **Analisi sperimentale di stimoli sonori non linguistici: individuazione dei parametri acustici significativi della risata**, Tesi di Laurea, Università di Milano, 1990.

W. F. Fry and C. Hader, "The respiratory components of mirthful laughter", **Journal of Biological Psycology**, 19, 1977, pp. 39-50.

S. Kori, "Perceptual dimensions of laughter and their acustic correlates and cultural diversity in the recognition of laughter content", **Bulletin of the Language Laboratory**, Osaka University of Foreign Studies, N. 8, 1985, pp. 21-48.

E.L. Lloyd, "The respiratory mechanism in laughter", **The Journal of General Psychology**, 19, 1938, pp. 179-189.

S.K. Langer, **Mind: an essay on Human feeling**, Hopkins University Press, Baltimore, 1973.

D.E. Mowrer, L.L. Lapointe, and J. Case, "Analysis of five acustic correlates of laughter", **Journal of Nonverbal Behaviour**, 11(3), 1987, pp. 191-199.

R. Plutchik and H. Kellerman, **Emotion: Theory, Research and Experience**, Academic Press, New York, 1980.

K.R. Scherer, "Methods of research on vocal communication: paradigms and parameters", in K.R. Scherer and P. Ekman **Handbook of methods in nonverbal behavior research**, Cambridge University Press, 1982, pp. 136-198.

H. Spencer, "The physiology of laughter", **Macmillan's Magazine**, 1, 1860, pp.395-402.

J. Sundberg, "Speech, song, and emotions", in M. Clynes (Ed.) **Music, mind and brain**, Plenum Press, London, 1981, pp. 137-150.

S. Svebak, "Respiratory patterns as predictors of laughters", **Psychophisiology**, Vol.12, N.1, 1975, pp. 62-65.

R. Van Bezooyen, **Characteristics and Recognizability of Vocal Expression of Emotion**, Foris Publications, Dordrecht, 1984.

J.A.R.A.M. Van Hooff, "Analisi comparata della filogenesi del riso", in R.A. Hinde (Ed.), **La comunicazione non verbale**, Laterza, Bari, 1974, pp. 277-318.

C.E. Williams and K.N. Stevens, "Emotions and speech: some acoustic correlates", **JASA**, 52(1), 1972, pp. 1238-1250.

Capitolo 3: "The Intelligent Musical Workstation"

Consiglio Nazionale delle Ricerche
Progetto Finalizzato Informatica
Sistemi Informatici e Calcolo Parallelo, LRC C4 MUSIC

Capitolo 3: "The Intelligent Musical Workstation"

Consiglio Nazionale delle Ricerche
Progetto Finalizzato Informatica
Sistemi Informatici e Calcolo Parallelo, LRC C4 MUSIC

ARCHITETTURA E AMBIENTI OPERATIVI DELLA STAZIONE DI LAVORO MUSICALE INTELLIGENTE°

A. Camurri^, G. Haus*

^ D.I.S.T .- Dipartimento di Informatica, Sistemistica e Telematica
Universita degli Studi di Genova, via Opera Pia 11a, I-16145 Genova, Italia
E-mail: music@dist.unige.it
Tel. +39 10 3532983 Fax +39 10 3532948

* L.I.M. - Laboratorio di Informatica Musicale, Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano, via Moretto da Brescia 9, I-20133 Milano, Italia
E-mail: music@imiucca.unimi.it
Tel: +39 2 7575249 Fax: +39 2 76110556

Parole chiave: computer music, audio digital signal processing, artificial intelligence, programming & microprogramming languages .

ABSTRACT

The "Intelligent Music Workstation" (IMW) is designed to be a music software/hardware environment in which commercially available products are integrated with prototypal modules developed in the frame of computer music research. The MIDI Management Tools standard is implemented; it allows the end-user to pipe the MIDI output of any software module as the MIDI input of any other software module within the IMW environment. Therefore, it is possible to have many programs which run concurrently while doing complex functions based on real-time mutual exchanges of MIDI data. The IMW software modules are capable to import/export Standard MIDI File 1.0 in all the available formats. Soundfiles are compatible with the Digidesign SoundDesigner II format. A hypertext musician/workstation interface is provided to enhance operative simplicity and evocative interaction. The IMW is an open environment, so that musicians can add their own applications without any troubles due to special formats of music data at both the MIDI and sound levels of representation. There are several specific environments for the end-user which can be briefly summed up by the following areas of user's activities: *sound editing*, *timbre programming*, *MIDI events programming*, *computer aided music composition*, *computer aided musicology*, *music knowledge representation*, *multimedia composition and performance*. A special subsystem of the IMW is the Sound Processing Environment (SPE); IMW-SPE integrates MIDI devices and DSP units allowing the user to control both MIDI devices and DSP units within the same sequencing environment so that DSP units may be considered as if they were real-time MIDI devices with timbre controlling capabilities#. The end-user has complete control over the whole environment by means of MIDI programming only and without the need of DSP microprogramming.

° Questa ricerca è parzialmente supportata dal Consiglio Nazionale delle Ricerche (CNR) nell'ambito della LRC C4: MUSIC (Contratti N° 89.00031.69 e 90.00678.PF69), "STAZIONE DI LAVORO MUSICALE INTELLIGENTE", Progetto C: SISTEMI AVANZATI DI PRODUTTIVITA' INDIVIDUALE, Sottoprogetto 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Progetto Finalizzato SISTEMI INFORMATICI E CALCOLO PARALLELO.

Patent pending.

1. Introduzione

L'attività di ricerca del nostro progetto si colloca nel contesto internazionale dell'informatica musicale come attività "ponte" tra la ricerca pura e l'applicazione industriale; con tale linea guida, si è inteso conseguire i seguenti obiettivi (Camurri et al., 1989a):

O1) realizzazione e sperimentazione di un'interfaccia utente evoluta (grafica spinta, linguaggi iconici, navigazione ipertestuale) per la Stazione di Lavoro Musicale Intelligente (SLMI);

O2) integrazione di funzionalità musicali in un'unica stazione di lavoro con possibilità di intervento da parte dell'utente ai vari livelli di rappresentazione dell'informazione musicale che caratterizzano i diversi ambiti operativi del settore;

O3) integrazione dei moduli applicativi commerciali con i moduli applicativi prototipali della SLMI;

O4) validazione della SLMI mediante attività pilota di utenze specifiche del settore.

Gli obiettivi prefissati sono orientati all'incremento della produttività musicale individuale e al conseguimento dell'autonomia operativa dei professionisti del settore musicale.

L'attività di ricerca della SLMI è condotta dalle due unità operative del Laboratorio di Informatica Musicale del Dipartimento di Scienze dell'Informazione dell'Università degli Studi di Milano e del Dipartimento di Informatica, Sistemistica e Telematica dell'Università degli Studi di Genova.

Nei successivi paragrafi viene fornita una descrizione ad alto livello dell'architettura e degli ambienti operativi della SLMI e dei singoli moduli software di cui è costituita.

In particolare, sono descritte:

* l'architettura software complessiva della SLMI;
* l'architettura hardware complessiva della SLMI;
* le funzionalità di alto livello che possono configurarsi come ambiti operativi specifici per l'utente finale, sia in relazione ai moduli propri della SLMI che in relazione a moduli applicativi reperibili in commercio;
* le funzionalità di alto livello dei singoli moduli software della SLMI;
* le librerie di modelli, dati e programmi sviluppate e di corredo per la SLMI;
* gli standard di codifica e di ambiente di sviluppo software adottati per il progetto SLMI.

2. Architettura della SLMI

La SLMI si configura come un insieme di moduli hardware e software organizzati in modo gerarchico. In prima approssimazione, il sistema è strutturato a cinque livelli:

Livello hardware:	-	Sistema di elaborazione general purpose
	-	Moduli per la elaborazione numerica di suoni
	-	Dispositivi per la registrazione e la riproduzione del suono
	-	Dispositivi di controllo e distribuzione di segnali (patchbay MIDI e di segnali audio, mixer)
Livello S/W 1:	-	Moduli a basso livello per le attività di analisi-sintesi del suono, la costruzione e l'elaborazione di campioni e modelli timbrici, la produzione di master audio numerici e il restauro di registrazioni audio

Livello S/W 2:	-	Moduli ad alto livello per le attività di composizione, analisi/sintesi di testi musicali e di strutture di testi musicali, performance musicale e audiovisuale, orchestrazione, DTP musicale
Livello S/W 3:	-	Moduli ad alto livello per la rappresentazione di conoscenza musicale
Interfaccia utente:	-	Interfaccia utente ad alto livello di tipo ipertestuale

Nello schema architetturale in Fig. 1 (Camurri et al., 1989b) sono rappresentati i livelli S/W 1, 2 e 3, mentre nel § 7. verrà descritto il livello hardware. Il livello "Interfaccia utente" consente la presentazione di oggetti/entità ad un livello di astrazione adatto ad un utente che intenda operare in un particolare sottoambiente, l'autoistruzione per l'uso della SLMI e la navigazione attraverso i particolari moduli applicativi con estrema flessibilità; la comunicazione visiva è a questo livello determinante ed è realizzata mediante le tipiche tecniche ad icone dei sistemi ipertestuali; il linguaggio iconico che caratterizza questo livello è sufficientemente generale da risultare idoneo per i vari tipi di utenti dei diversi ambiti operativi che possono essere fruiti. Le frecce che collegano i moduli della SLMI nello schema indicano i collegamenti funzionali possibili tra i moduli (sia prototipali che commerciali): i risultati in uscita da un modulo possono infatti costituire i dati in ingresso per un altro modulo (vedi § 6., pipeline di moduli applicativi). Nella colonna a destra nello schema sono indicati i tipi di moduli applicativi reperibili in commercio che sono integrati con la SLMI.

3. Ambienti operativi della SLMI

Rispetto ai prodotti attualmente disponibili sul mercato internazionale, la SLMI costituisce un nuovo tipo di prodotto, pur nella sua veste prototipale, in cui le funzionalità usualmente separate sono integrate in un unico e più accessibile ambiente di lavoro; l'effetto principale è un sensibile incremento di produttività, considerando la stazione di lavoro tanto nella sua completezza (eventualmente in differenti versioni corrispondenti a vari livelli di prestazioni), quanto considerando i suoi ambienti operativi specifici messi a disposizione dall'interfaccia ipertestuale; gli ambienti predisposti sono orientati alla composizione musicale, alla performance musicale e multimediale, all'analisi/sintesi di partiture, all'orchestrazione, al laboratorio audio digitale, al mastering di supporti audio, al browsing di Standard MIDI Files e al browsing di Soundfiles.

Inoltre, la SLMI è un ambiente aperto a cui si possono aggiungere e togliere moduli applicativi e quindi organizzarli nei più svariati modi per rendere più efficienti le modalità di utilizzo della SLMI stessa da parte dell'utente finale. In pratica la SLMI ha un suo insieme di moduli applicativi propri, una ventina, che possono essere utilizzati in una qualsivoglia combinazione e integrati con un qualsivoglia insieme di moduli applicativi commerciali per costituire ambiti operativi specifici, realizzabili direttamente dall'utente mediante gli strumenti messi a disposizione dall'interfaccia ipertestuale. Questi nuovi ambienti si aggiungeranno o si sostituiranno a quelli già disponibili nel nostro prototipo.

In questo paragrafo sono brevemente discusse le funzionalità di alto livello della SLMI che possono configurarsi come ambiti operativi specifici per l'utente finale, sia in relazione ai moduli propri della SLMI che in relazione a moduli applicativi reperibili in commercio.

Rappresentazione di conoscenza musicale
Uno degli obiettivi primari della SLMI è quello di assistere il compositore nelle fasi altamente creative del processo compositivo. Ciò include la definizione di piani e architetture per una composizione e la

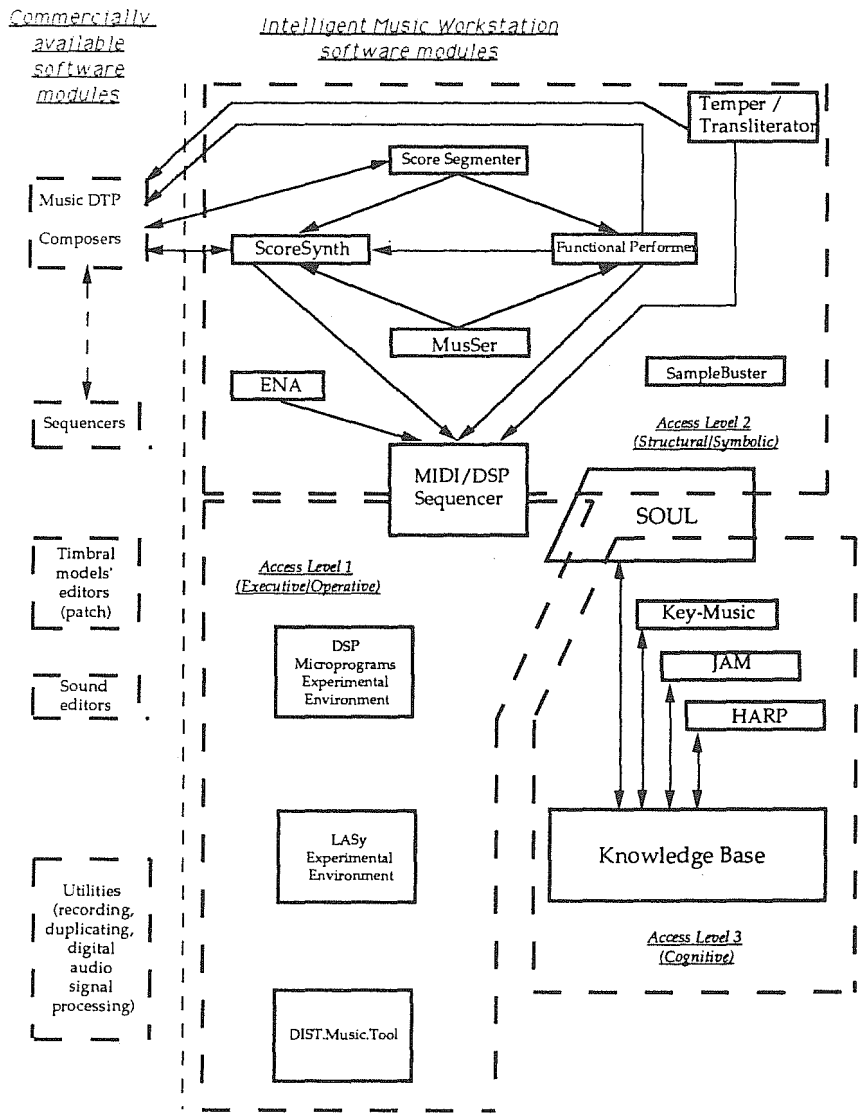

Commercially available software modules
Intelligent Music Workstation software modules
Temper / Transliterator
Score Segmenter
Music DTP
Composers
ScoreSynth
Functional Performer
MusSer
ENA
SampleBuster
Sequencers
Access Level 2 (Structural/Symbolic)
MIDI/DSP Sequencer
SOUL
Timbral models' editors (patch)
Access Level 1 (Executive/Operative)
Key-Music
DSP Microprograms Experimental Environment
JAM
Sound editors
HARP
LASy Experimental Environment
Knowledge Base
Utilities (recording, duplicating, digital audio signal processing)
Access Level 3 (Cognitive)
DIST.Music.Tool

Figura 1

codifica del materiale musicale in una partitura operativa. Allo scopo di facilitare l'interazione col sistema, viene supportata una definizione più ampia possibile di "materiale musicale", mediante diverse rappresentazioni della conoscenza musicale. I sistemi *Key-Music* ed *HARP* (Hybrid Action Representation and Planning) hanno l'obiettivo di fornire la soluzione per alcuni di questi problemi. In particolare, Key-Music è un sistema per la rappresentazione di conoscenza musicale basato su reti di Petri e reti semantiche di frame ad eredità multipla; mentre HARP consiste in un sistema ibrido, ovvero integra aspetti di tipo simbolico ed analogico. Ad essi verrà affiancato il modulo *JAM* , un sistema prototipale per la rappresentazione di conoscenza musicale. Un altro obiettivo primario della SLMI è quello di fornire ai ricercatori strumenti di indagine conoscitiva su vari aspetti dell'ambito musicale. Tra questi ultimi, uno dei più rilevanti riguarda le problematiche dell'ascolto. A tale proposito verrà realizzato il modulo *SOUL* (Self-Organizing Universal Listener), implementazione di un modello per l'*ascolto intelligente* , ovvero per il riconoscimento di una sorgente sonora (tipicamente uno strumento musicale) a partire da un segnale complesso (costituito ad esempio da più strumenti). Quest'ultimo modello è basato su reti neurali, in particolare sul modello di Kohonen.

Composizione

L'attività di composizione musicale può anche essere realizzata combinando la scrittura tradizionale resa possibile dai programmi *composer* commerciali insieme con la scrittura non tradizionale basata su reti di Petri, oggetti musicali e algoritmi con il modulo *ScoreSynth*. Per la composizione di opere musicali di natura seriale verrà realizzato *MusSer*, un modulo specifico per la generazione di serie musicali equilibrate con un qualsivoglia temperamento equabile a n gradi (con n pari). Gli oggetti musicali argomento degli algoritmi definiti nei modelli ScoreSynth possono essere derivati tanto da moduli applicativi commerciali via Standard MIDI File che dal modulo MusSer o anche dal modulo *ScoreSegmenter* che decompone in oggetti musicali intere partiture precedentemente codificate.

Analisi/sintesi testi musicali

La trasformazione di testi musicali preesistenti è una peculiare funzionalità della SLMI resa possibile dall'uso combinato dei moduli *ScoreSegmenter* (per l'analisi della partitura e la sua decomposizione in oggetti musicali), *ScoreSynth* (per la sintesi di trasformazioni della partitura mediante modelli di reti di Petri) e *Functional Performer* (per la sintesi di trasformazioni della partitura mediante operatori in tempo reale).

Analisi strutture testi musicali

Il modulo *ScoreSegmenter,* come già accennato qui sopra, consente di identificare gli oggetti musicali di cui è composta una partitura con una particolare affidabilità per i brani musicali in forma di fuga o di sonata. E' quindi uno strumento essenziale e fortemente innovativo di supporto al musicologo e in generale allo studioso dei testi musicali.

Performance musicale

Il modulo *Functional Performer* permette di eseguire brani musicali attivando da tastiera ASCII funzioni musicali che vengono applicate ad oggetti musicali predisposti precedentemente o tramite moduli SLMI o tramite composer o sequencer commerciali in formato Standard MIDI File. Il modulo *ENA* (Experimental Neural Accompanist), basato su reti neurali di tipo back-propagation, genera in tempo reale un accompagnamento musicale automatico durante l'esecuzione di una parte solistica.

Performance audiovisuale

Il modulo *TEMPER* permette di eseguire automaticamente processi audiovisuali costituiti da animazioni grafiche nello stile del pittore M. C. Escher e processi MIDI ottenuti per traslitterazione delle animazioni grafiche (intepretate come piani melodici).

Analisi-sintesi suono
E' un ambiente sperimentale in cui è disponibile un modulo (microprogramma per il DSP Motorola 56001) per l'analisi-sintesi con la trasformata wavelet. In combinazione con programmi come Digidesign SoftSynth e Digidesign SoundDesigner permette di confrontare e potenziare le caratteristiche della analisi-sintesi additiva per armoniche con quelle della analisi-sintesi mediante trasformata wavelet.

Editing di campioni sonori
Il modulo DIST.Music.Tool della SLMI e il programma Digidesign SoundDesigner consentono di effettuare l'editing di campioni sonori direttamente all'elaboratore e di effettuare operazioni di import/export dalle periferiche MIDI a campionamento.

Preparazione modelli timbrici
E' un ambiente sperimentale in cui è disponibile un insieme di moduli (microprogrammi per il DSP Motorola 56001) per la sintesi del suono secondo vari modelli di sintesi: funzioni di due variabili, waveshaping, automi cellulari.

Orchestrazione
L'attività di orchestrazione è una delle più complesse per il musicista; richiede un ciclo ipotesi-di-orchestrazione/sperimentazione-sonora/editing-dell'assetto-orchestrale che si ripete tante volte quante sono necessarie per il raggiungimento dell'obiettivo di tessitura timbrica ricercata; per questo è stato progettato un ambiente in cui siano operativi concorrentemente due tipi di moduli software: uno di supporto alla ricerca dei suoni nella banca di timbri disponibili (il modulo *SampleBuster* o gli ambienti per la sperimentazione timbrica nella SLMI) e l'altro per l'esecuzione dei brani nelle varie ipotesi di tessitura timbrica (un qualunque sequencer MIDI commerciale o il modulo *Sequencer MIDI-DSP* nella SLMI, che consente di controllare contemporaneamente unità di sintesi MIDI e DSP microprogrammati in tempo reale).

DTP musicale
Integrando i moduli della SLMI capaci di generare partiture (*ScoreSynth, Functional Performer*) con gli strumenti per il Desk Top Publishing musicale commerciali (ad es. il programma Finale della Coda Music Software), è possibile ottenere le partiture generate in notazione tradizionale e ai massimi livelli di qualità (fotocompositrici che accettano codici PostScript; es. Linotronics 100 P).

Produzione master audio numerici
I programmi sequencer commerciali, il modulo *Sequencer MIDI-DSP* e alcuni programmi e dispositivi di utilità (Digidesign DAT I/O, Digidesign SoundTools) consentono di registrare, prima a livello MIDI e poi come audio digitale su disco rigido SCSI, su nastro DAT e come audio analogico su nastri open reel, il master delle produzioni musicali.

Restauro registrazioni audio
Il modulo DIST.Music.Tool della SLMI e il programma Digidesign SoundDesigner consentono di effettuare l'editing di registrazioni audio per il restauro di vecchi supporti; anche in questo caso, alcuni programmi e dispositivi di utilità (Digidesign DAT I/O, Digidesign SoundTools) consentono di spostare da un supporto all'altro e di convertire nei formati richiesti l'informazione audio digitale tra nastri DAT, dischi rigidi SCSI, campionatori MIDI, campionatori SCSI, nastri analogici open reel.

4. Librerie disponibili nella SLMI

I moduli della SLMI hanno accesso in lettura e scrittura a un insieme di librerie che costituiscono uno degli elementi costitutivi essenziali della SLMI stessa; in questo paragrafo vengono succintamente

descritte tali librerie; in (Stiglitz Ed., 1991) sono dettagliati i vincoli relativi agli accessi per ogni singolo modulo.

Libreria di reti di Petri "macro"
Il modulo *ScoreSynth* ha accesso ad una libreria di reti di Petri "macro" ovvero un insieme di costrutti di reti di Petri che sono descritti in forma parametrica e che possono essere invocati mediante parametri formali per valore. Questa libreria è stata progettata per rendere disponibile quell'insieme di costrutti che ricorrono sovente nello sviluppo di modelli di partiture con reti di Petri e rendere perciò più efficiente la fase di editing dei modelli formali.

Libreria di routine di lettura/scrittura di Standard MIDI File v 1.0
Questa libreria (costituita da una cinquantina di routine) costituisce un vero e proprio modulo della SLMI e come tale sarà descritta in R3 e più succintamente nel prossimo paragrafo. E' stata progettata per rendere disponibile in forma standard a tutti i ricercatori del presente progetto il principale strumento di comunicazione con i moduli applicativi reperibili in commercio e tra i moduli della SLMI stessi. Tanto le applicazioni commerciali quanto i moduli della SLMI hanno funzionalità di import/export di file a livello di partitura MIDI come Standard MIDI File v 1.0.

Libreria di patch timbriche per microprogrammi del DSP Motorola 56001
Il modulo *Sequencer MIDI-DSP* consente di variare parametricamente il modello timbrico corrente mediante patch timbriche; ogni microprogramma sviluppato per tale modulo, come anche i microprogrammi per l'*Ambiente sperimentale per microprogrammi DSP* , avranno a disposizione un corredo di patch timbriche (analogamente a quanto accade per i preset dei sintetizzatori MIDI) organizzati in una libreria accessibile dai due moduli citati. L'utente di tali moduli potrà ovviamente sviluppare le sue specifiche patch timbriche e aggiungerle alla libreria sviluppata nell'ambito del presente progetto.

Banca timbrica di campioni sonori
Il modulo *SampleBuster* è basato su una banca timbrica di campioni sonori costituita da circa 7.000 elementi delle seguenti famiglie timbriche originali:
f1) MUMS (McGill University Master Samples)
f2) Sound Examples (Current Directions in Computer Music Research, M.I.T.)
f3) AKAI S900 Samples Library
f4) AKAI S950 Samples Library
f5) AKAI S1000 Samples Library
f6) libreria di suoni campionati del L.I.M.
le famiglie timbriche f1 ed f2 sono memorizzate su CD-DA (Compact Disc Digital Audio); le famiglie f3, f4, f5 ed f6 sono memorizzate su dischi magnetici.

5. Descrizione sintetica dei moduli della SLMI

In questo paragrafo vengono descritti sinteticamente tutti i moduli sperimentali che fanno parte dell'architettura della SLMI. Per una descrizione più dettagliata e formale delle funzionalità dei singoli moduli, si rimanda a (Stiglitz Ed., 1991).

ScoreSynth
Il modulo **ScoreSynth** consiste in un'applicazione, sviluppata in ambiente MPW su computer Macintosh, che permette di sintetizzare partiture musicali mediante l'esecuzione di modelli formali sviluppati con reti di Petri. I modelli che utilizzati consistono in un particolare adattamento delle reti di Petri al contesto musicale, e in un'algebra musicale. Gli *oggetti musicali* sono associati a *posti* e sono codificati in uno dei seguenti tre formati: un codice musicale alfanumerico, Standard MIDI File v.1.0,

una codifica numerica interna dei dati MIDI. Le funzioni musicali sono descritte per mezzo di espressioni associate a *transizioni* e sono codificate mediante operatori dell'algebra musicale appositamente definita. Gli operatori possono essere applicati ai parametri musicali *altezza*, *durata*, *dinamica* (intensità); inoltre possono modificare l'*ordinamento* degli oggetti musicali. L'approccio seguito permette di descrivere gli oggetti musicali e le loro trasformazioni all'interno delle composizioni in modo svincolato dal livello simbolico di rappresentazione che è tipico della comune notazione musicale. In questo modo la musica appare come un modello multistratificato nel quale il musicista sceglie il numero e le caratteristiche dei diversi livelli di rappresentazione.

MusSer

Il modulo **MusSer** consiste in un'applicazione, sviluppata in ambiente MPW su computer Macintosh, che permette di generare pattern seriali equilibrati in ambito di scale temperate di cardinalità pari. Le serie equilibrate sono una delle possibili strutture base utilizzate nella composizione di musica seriale. Utilizzando MusSer, un compositore è in grado di valersi del computer nella fase di sviluppo e ricerca dei pattern basilari per una composizione seriale. Infatti ha a disposizione archivi permanenti, residenti su memoria di massa, contenenti tutte le sequenze numeriche corrispondenti alle serie dodecafoniche di base, e gli strumenti per la ricerca selettiva delle serie con le caratteristiche desiderate, e per la loro elaborazione. In questo modo può creare archivi contenenti pattern seriali "personalizzati" da poter utilizzare con altri moduli di ausilio alla composizione musicale appartenenti alla SLMI.

Sequencer MIDI/DSP

Il modulo **Sequencer MIDI/DSP** consiste in un'applicazione, sviluppata in ambiente MPW su computer Macintosh, di tipo *sequencer* (registratore e processore di sequenze di dati musicali) per il controllo simultaneo e sincrono di unità MIDI e di schede DSP (brevetto depositato in Italia) e per il mixing a livello MIDI di sequenze musicali. L'applicazione è composta da due sezioni: un *sequencer MIDI* che unisce alle consuete funzioni di registrazione, riproduzione ed elaborazione di sequenze MIDI anche la possibilità di controllare, tramite un apposito driver, la scheda Digidesign Sound Accelerator (basata sul DSP Motorola 56001) collocata internamente al Macintosh, vedendola come una normale unità di sintesi e/o campionamento esterna controllabile via MIDI; un *mixer* per il bilanciamento dei livelli di intensità di differenti e contemporanee sequenze di dati musicali (numero massimo: 32) che opera direttamente sulle sequenze di dati MIDI, agendo sul parametro MIDI standard *key-velocity* , ed eliminando quindi la necessità di utilizzare per l'attività di mixing un apposito dispositivo hardware esterno di tipo convenzionale.

Microprogrammi per Sequencer MIDI-DSP e Sound Accelerator Card

Il modulo **Microprogrammi per Sequencer MIDI-DSP e Sound Accelerator Card** consiste in una libreria di microprogrammi, sviluppata in ambiente MPW su computer Macintosh con scheda Digidesign Sound Accelerator (basata su DSP Motorola 56001), per la sintesi del suono mediante DSP dedicato. I microprogrammi implementano le seguenti teniche di sintesi: TVF (funzioni a due variabili), Waveshaping, Wavelet e Automi Cellulari; e sono utilizzabili in ambiente MPW (ad esempio vengono usati, nell'ambito della SLMI, per lo sviluppo dell'*Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card*).

Functional Performer

Il modulo **Functional Performer** consiste in un'applicazione, sviluppata in ambiente MPW su computer Macintosh, che permette di realizzare performance funzionali in tempo reale di pattern musicali e loro trasformazioni. L'applicazione accetta in input frammenti musicali (melodici polifonici) codificati in due formati: Standard MIDI File v.1.0 generati da altre applicazioni della SLMI o commerciali; formato di rappresentazione interna. I frammenti possono essere organizzati in appositi archivi caricabili all'inizio di una performance. L'utente-esecutore ha quindi a disposizione un certo numero di operatori, attivabili agendo col mouse o con la tastiera ASCII del Macintosh, per la trasformazione e l'esecuzione

in tempo reale dei frammenti musicali originari e/o derivati. Gli operatori implementano una serie di trasformazioni tipiche della prassi compositiva e improvvisativa moderna (trasposizioni, inversioni speculari, retrogradazioni, ecc.), che agiscono sui parametri altezza e durata degli elementi dei frammenti musicali. L'esecuzione provoca la generazione di sequenze di dati MIDI per il controllo di unità esterne di sintesi e campionamento del suono, collegate al Macintosh mediante interfaccia MIDI. Le sequenze possono anche essere registrate, durante l'esecuzione, in archivi in formato Standard MIDI File v.1.0. Durante l'esecuzione è visibile sullo schermo del Macintosh una partitura grafica a scansione temporale, con la visualizzazione dei frammenti musicali attivi, distribuiti in varie fasce orizzontali corrispondenti ai canali MIDI utilizzati.

TEMPER

Il modulo **TEMPER** (TEssellating Music PERformer) consiste in un'applicazione, sviluppata in ambiente Macintosh, che permette di generare performance audiovisuali basandosi su tassellazioni del piano nello stile dell'artista grafico olandese M.C.Escher. L'applicazione è composta da due sezioni: un *editor* che consente di costruire, memorizzare e modificare sequenze di tassellazioni (una tassellazione è una partizione del piano in insiemi connessi di punti; in altri termini, il piano viene suddiviso in più figure senza "vuoti", ovvero con i contorni combacianti) appartenenti a una classe particolare (gruppo spaziale bidimensionale P1); ed un *performer* che esegue trasformazioni grafiche delle tassellazioni indicate dall'utente, accompagnate eventualmente dalla corrispondente esecuzione musicale. Per quanto riguarda quest'ultima, sono generate sequenze di dati MIDI utilizzabili per controllare direttamente unità esterne per la sintesi e/o il campionamento del suono, o per essere registrate da un programma di tipo "sequencer". La generazione delle sequenze di dati MIDI avviene in tempo reale e produce eventi musicali sincronizzati con le trasformazioni grafiche, secondo la forma dei contorni delle figure e alcuni parametri specificati dall'utente (velocità di esecuzione, scala musicale, ecc.).

ScoreSegmenter

Il modulo **ScoreSegmenter** consiste in un'applicazione, sviluppata in ambiente Macintosh, che permette di segmentare partiture date in oggetti musicali. La segmentazione di brani musicali è da considerarsi quale primo passo per una futura loro strumentazione automatica proposta dall'elaboratore. Si tratta cioé della ricerca dei vari oggetti musicali di cui é composto il brano, dove, in questo contesto, per oggetti si intendono quei fraseggi o pensieri musicali che l'autore ha espresso e che ha ripreso e trasformato, secondo i vari canoni musicali, in relazione al periodo storico e alla forma compositiva. La realizzazione di ScoreSegmenter rappresenta il primo tentativo di costruire uno strumento informatico che rende possibile l'accostarsi ai testi musicali non come mera sequenza di note, cioé di codifica di eventi sonori, come la pratica strumentale richiede, ma considerandoli maggiormente come impianti strutturali di una particolare forma espressiva (quella del compositore) che si reggono su elementi base anche molto complessi (materiale tematico e armonico di impianto). Gli algoritmi per la ricerca delle occorrenze degli oggetti musicali e per la successiva segmentazione di un brano musicale, fanno specificamente riferimento alla fuga e alla forma sonata per poi essere successivamente generalizzati ad altre forme musicali mediante "esperienze" del sistema.

SampleBuster

Il modulo **SampleBuster** consiste in un sistema ipertestuale, sviluppato in ambiente HyperCard su computer Macintosh, che permette di selezionare campioni sonori, appartenenti ad una banca di 7000 suoni, sulla base di attributi multipli. Il sistema si articola in più stack, attivi in ambiente HyperCard, che consentono di effettuare le seguenti attività: controllo diretto dei campionatori Akai S950 e S1000, collegati al Macintosh via MIDI, per il trasferimento (modalità MIDI *Sistema Esclusivo*) e l'editing di campioni sonori e di patch; consultazione di una biblioteca di informazioni sugli strumenti musicali, sia di carattere organologico, sia riguardanti le possibilità di collocazione degli strumenti stessi in organici per l'esecuzione di musiche appartenenti a generi diversi (dalla musica sinfonica e operistica, al jazz e al rock); ricerca selettiva, sulla base di attributi multipli selezionabili dall'utente, di campioni sonori

appartenenti ad una banca costituita da diverse collane di suoni, tra cui Akai "Workstation" e CD McGill University "Master Samples", distribuite su supporti differenziati (floppy disk per campionatori, CD-DA).

ENA

Il modulo **ENA** (Experimental Neural Accompanist), sviluppato in ambiente Macintosh, consiste in un'applicazione per la generazione in tempo reale di accompagnamenti in corrispondenza dell'esecuzione di parti musicali solistiche. Il modulo è basato su reti neurali di tipo "back propagation" che vengono istruite mediante un insieme di melodie tonali, con relativi accompagnamenti e armonizzazioni, di autori classici del Settecento. Il solista può utilizzare per l'esecuzione una tastiera MIDI collegata al Macintosh oppure la tastiera ASCII. Il modulo si presta particolarmente per applicazioni nel campo della valutazione delle performance musicali o per l'esecuzione diretta di composizioni ad hoc.

DIST.Music.Tool

Il modulo **DIST.Music.Tool** consiste in un ambiente (programmi e librerie) per la generazione e l'editing di campioni sonori. In particolare, DIST.Music.Tool è costituito dai seguenti componenti: *STED* (programma per l'editing di campioni); *AKAIComm* (programma per la gestione della comunicazione PC-campionatore Akai S900); *HDMusic Toolkit* (libreria di routine e programma "monitor" per la ricezione/trasmissione in DMA di campioni sonori via ADC/DAC per mezzo dell'apposita scheda compresa nel kit e montata nel PC, e per la scrittura/lettura su hard disk dei dati corrispondenti). Inoltre, sempre nell'ambito DIST.Music.Tool, sono stati portati in ambiente PC, adattati ed espansi, il modulo *CARL* (sviluppato all'Università della California di S.Diego) contenente il linguaggio per la composizione musicale C-Music, e il modulo*CMU MIDI Toolkit* (sviluppato all'Università Carnegie-Mellon) per la gestione di campioni in ambiente MIDI (mediante l'interfaccia MIDI per PC Roland MPU-401).

Rappresentazione della conoscenza musicale: Key-Music, HARP, JAM

I moduli **Key-Music** ed **HARP** (Hybrid Action Representation and Planning) hanno l'obiettivo di fornire la soluzione per alcuni dei problemi più significativi legati alla rappresentazione di conoscenza in ambito compositivo. In particolare, Key-Music è un sistema per la rappresentazione di conoscenza musicale basato su reti di Petri e reti semantiche di frame ad eredità multipla; HARP consiste invece in un sistema ibrido, ovvero che integra aspetti simbolici e analogici. Entrambi traggono i loro fondamenti dal fatto che, in musica, il **tempo** gioca un ruolo fondamentale: un fenomeno musicale (ad esempio l'esecuzione di un brano) può essere rappresentato mediante attori musicali interagenti e temporizzati (l'analogia orchestra/direttore con processi concorrenti/scheduler è, seppur banale, evocativa dell'importanza di tale aspetto nella rappresentazione). Key-Music è basato su un modello a reti semantiche per la rappresentazione della **informazione descrizionale**, e sulle reti di Petri per la **informazione fattuale** su attività concorrenti e temporizzate. Verrà utilizzato un sistema basato su reti semantiche e frame per rappresentare la conoscenza che concerne definizioni generali su processi musicali temporizzati, sotto forma di reti di frame; in particolare, verranno sfruttate le capacità di inferenza caratteristiche di tale formalismo: **classificazione ed ereditarietà**. Rappresentazioni di processi individuali o attuali possono essere automaticamente generate come **istanze** di una definizione generale, in termini di **reti di Petri**. Tale modello è alla base di un modulo della stazione di lavoro dedicato all'ausilio alla composizione ed alla didattica. Una implementazione prototipale è in corso di realizzazione, utilizzando il linguaggio Lisp (CLOS - Common Lisp Object System) su workstation Unix/X-Window. Le informazioni contenute in un tale tipo di base di conoscenza sono la rappresentazione astratta simbolica in grado di generare **automaticamente** molte specifiche istanze in termini di reti di Petri: tale modello è stato denominato **A-Net** (Action Nets). Le A-Net possono essere viste come una "superclasse" di reti di Petri ad un livello di astrazione superiore. A questo livello elaboriamo oggetti più astratti sui quali possiamo intervenire in due modi: derivando particolari oggetti individuali (reti di Petri) o ragionando attorno alle loro proprietà come in qualsiasi sistema di

rappresentazione della conoscenza. Ad esempio, possiamo dire che il livello più alto consente di descrivere **azioni** di tipo generale quali "comporre una fuga" o "improvvisare su un tema"; il corrispondente livello di rete di Petri (il più basso) è, ad esempio, in questo caso "componi una fuga a 3 voci, in Re maggiore, utilizzando il soggetto X ed il controsoggetto Y" o "improvvisa sul tema X...". La metodologia descritta è alla base di un sistema prototipale, denominato **JAM**, per la rappresentazione di conoscenza musicale. In tale prototipo verrà introdotta conoscenza riguardante armonia funzionale, pattern melodici e ritmici, riguardante un ben determinato stile musicale. Il sistema dovrà essere in grado di generare (in modo automatico o guidato) improvvisazioni su schemi armonici fissate, eventualmente interagendo con l'utente in tempo reale in tutte le fasi della creazione. La base di conoscenza verrà strutturata in forma di Action-Nets.

SOUL

L'obiettivo dell'ambito di ricerca del modulo **SOUL** (Self-Organizing Universal Listener) è lo studio preliminare di un sensore acustico intelligente adattivo per il riconoscimento e l'inseguimento di una sorgente sonora, basato sull'impiego di modelli connessionistici. In ambito musicale e psico-acustico, è nota la capacità biomorfa di estrarre la singola sorgente sonora (lo strumento) da un background non necessariamente definibile in termini di grandezze tipiche della teoria dei segnali (frequenze, intensità, correlazione, ecc.). Si tratta di una attività intelligente (coinvolge livelli di astrazione) e adattiva (basata sull'apprendimento). La tecnologia del sensore periferico è già ben consolidata e disponibile a basso costo (il trasduttore elettroacustico), mentre è da investigare la parte intelligente del sistema sensoriale, per cui la tecnologia corrente (elaborazione digitale su h/w dedicato, sistemi basati su conoscenza, data fusion) non è soddisfacente. La ricerca si basa sull'impiego di reti neurali; il modello che più sembra significativo è quello delle **self-organizing network**, sviluppato da Kohonen. Tale tipo di reti neurali dovrebbe consentire, dopo opportuno training, la formazione di aggregazioni di unità accordate a particolari pattern. Il modulo prototipale SOUL consiste di un sistema di simulazione integrato alla acquisizione e pre-processing del suono, e ad una interfaccia grafica per il monitoraggio della rete. Il modello dell'ascoltatore è formalizzato.

Routine di lettura/scrittura di Standard MIDI File v 1.0

Il modulo **Routine di lettura/scrittura di Standard MIDI File v.1.0** consiste in una libreria di routine, sviluppata in ambiente MPW su computer Macintosh, che permettono di leggere e scrivere archivi di dati MIDI secondo le specifiche fornite dalla *International MIDI Association.*. Le routine sono utilizzabili in ambiente MPW, e consentono di svolgere le funzioni inerenti la scrittura/codifica e la lettura/decodifica di Standard MIDI File (versione 1.0) nei formati 0 e 1, secondo le specifiche fornite dalla *International MIDI Association.* In particolare, sono state implementate routine per la codifica e la decodifica di numeri espressi in forma di "variable-length quantity" (rappresentazione "compressa" di valori numerici utilizzata negli SMF), routine per la lettura e la scrittura di header generali e header di traccia, routine di lettura/decodifica e codifica/scrittura di metaeventi, routine di lettura/decodifica e codifica/scrittura di eventi, routine di supporto per la diagnostica.

Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card

Il modulo **Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card**, sviluppato in ambiente MPW su computer Macintosh con scheda Digidesign Sound Accelerator (basata su DSP Motorla 56001), consente di costruire e modificare modelli timbrici basati su alcune tecniche specifiche di sintesi del suono (Wavelet, Waveshaping, TVF, Automi cellulari). In particolare, è costituito da quattro sottoambienti, uno per ciascuna tecnica di sintesi implementata: uno per l'editing di modelli timbrici basati sulla tecnica di sintesi per Wavelet; un altro per l'editing di modelli timbrici basati sulla tecnica di sintesi per Waveshaping; un altro ancora per l'editing di modelli timbrici basati sulla tecnica di sintesi per funzioni a due variabili; infine l'ultimo, *LASy* (Linear Automata Synthesis) per l'editing di modelli timbrici basati su una tecnica di sintesi che utilizza automi cellulari come modelli per l'evoluzione dei suoni. I modelli timbrici generati per mezzo delle quattro tecniche di sintesi possono

essere archiviati in formato Soundfile Digidesign per poi essere utilizzati da altri moduli della SLMI (ad es. *Sequencer MIDI/DSP*) o dalle applicazioni Digidesign (ad es. SoftSynth).

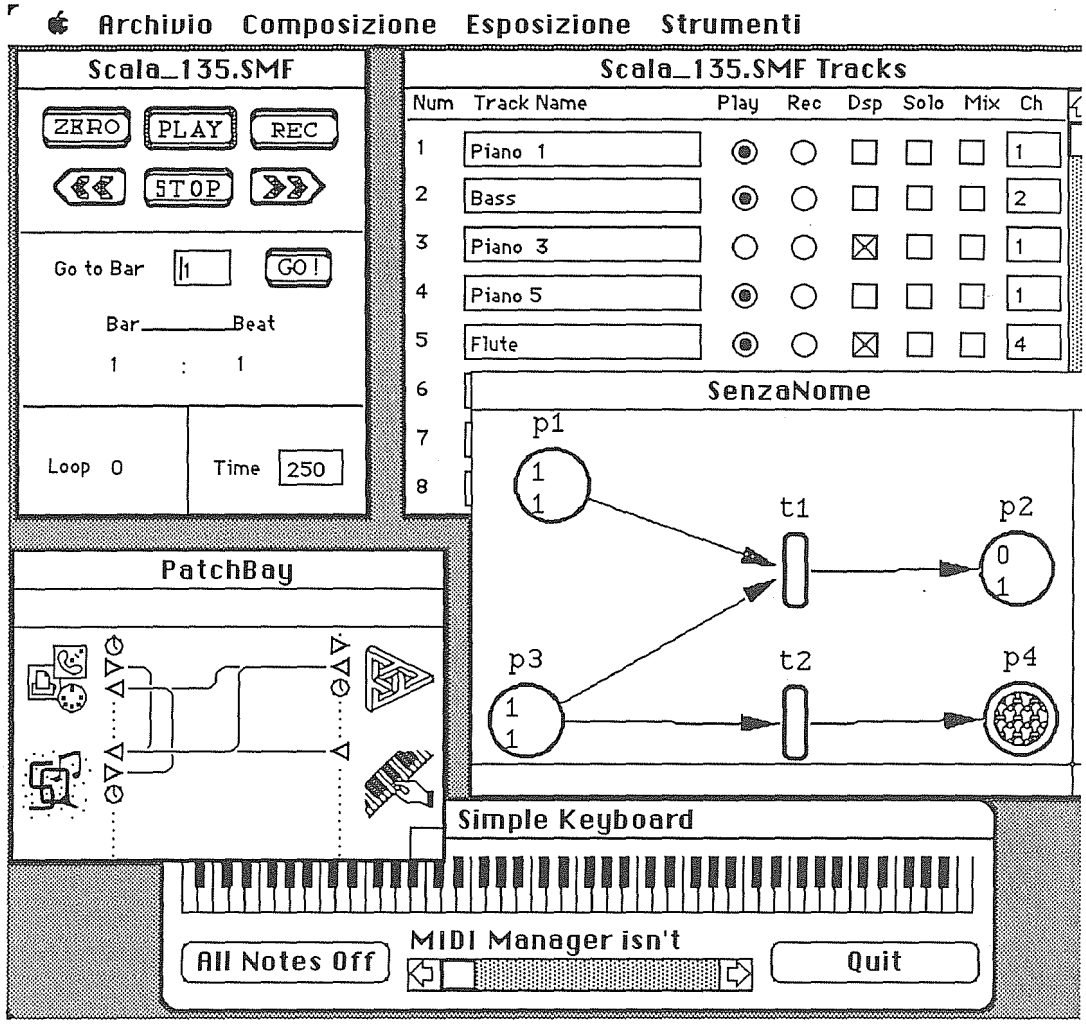


Figura 2

6. Standard utilizzati nell'implementazione

Al fine di rendere i moduli della SLMI integrati pienamente con i moduli applicativi reperibili in commercio è stato convenuto di adottare un insieme di standard per la codifica dei dati e per lo sviluppo del software di cui i seguenti sono i più rilevanti:

s1) *MIDI Management Tools 2.0* : come ambiente standard per lo sviluppo di moduli che utilizzano l'interfaccia MIDI sotto Multifinder (ambiente multitasking per Macintosh); consente di connettere logicamente in tempo reale i flussi MIDI in ingresso e in uscita ai/dai vari moduli della SLMI e dei moduli commerciali implementati nel medesimo ambiente; in tal modo i vari moduli si scambiano flussi di dati in tempo reale consentendo così di costruire applicazioni più complesse combinando opportunamente i singoli moduli;

s2) *Standard MIDI File v.1.0* : come standard per la codifica dei file contenenti partiture a livello MIDI; consente di scambiare tali informazioni tra tutti i moduli della SLMI e con i moduli commerciali che operano a livello di partitura MIDI;

s3) *Digidesign Soundfile* : come standard per la codifica di dati di tipo sonoro (campioni sonori, master digitali); consente di scambiare dati di tipo sonoro tra tutti i moduli di editing, di registrazione, di sintesi e di analisi del suono in forma numerica.

In Fig. 2 è illustrato un esempio di integrazione tra diversi moduli applicativi della SLMI in un unico modulo più complesso, con un metodo di connessione funzionale tipo *pipeline* : sia il modulo ScoreSynth che la tastiera MIDI fungono da generatori di eventi MIDI, il Sequencer MIDI/DSP registra gli eventi MIDI su tracce separate e ne consente l'ascolto inviando i codici MIDI alle periferiche per la sintesi del suono connesse alla interfaccia MIDI (fisica): tutti questi processi sono attivi concorrentemente in multitasking e l'operazione è resa possibile dall'implementazione nei moduli della SLMI dei MIDI Management Tools.

7. Configurazione hardware.

Nei diagrammi delle figure 3 e 4 sono rappresentate le configurazioni hardware che supportano i livelli S/W (1, 2, 3) di accesso alla SLMI.

8. Ambiti applicativi

Il prototipo di SLMI è da ritenere di interesse per una varietà di utilizzatori professionali e industriali tra cui: industrie di strumenti musicali elettronici; software house specializzate in applicazioni musicali e audio; studi di produzione musicale; studi di postproduzione video; editori musicali; editori multimediali; produttori di supporti ottici (CD-DA, CD-ROM, CD-I); industrie di elaboratori elettronici; compositori; arrangiatori; interpreti musicali; musicologi.

E' altresì da ritenere fortemente innovativo sia per le soluzioni tecniche e tecnologiche adottate che per le caratteristiche di flessibilità, di integrazione software e di interfaccia utente evoluta. In particolare, costituisce un salto di qualità rispetto alla produzione industriale giapponese e americana del settore (attuali leader del mercato) tipicamente specializzata a funzioni particolari e caratterizzata da prodotti con funzionalità molto frammentate e difficilmente integrabili tra loro.

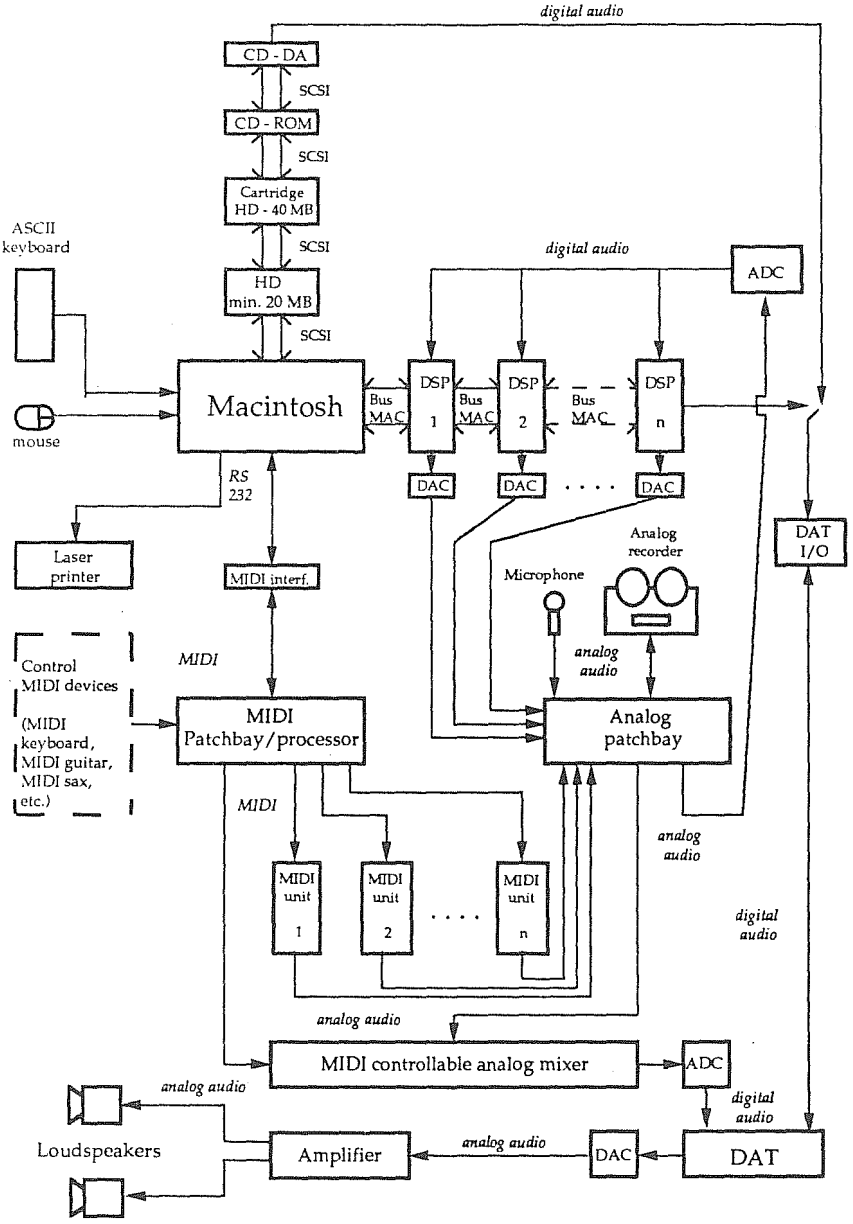

digital audio
CD - DA
SCSI
CD - ROM
SCSI
Cartridge
HD - 40 MB
SCSI
HD
min. 20 MB
SCSI
ASCII
keyboard
mouse
Macintosh
RS
232
Laser
printer
MIDI interf.
Control
MIDI devices
(MIDI
keyboard,
MIDI guitar,
MIDI sax,
etc.)
MIDI
MIDI
Patchbay/processor
MIDI
unit
1
MIDI
unit
2
MIDI
unit
n
Bus
MAC
DSP
1
DSP
2
DSP
n
DAC
DAC
DAC
digital audio
ADC
DAT
I/O
Analog
recorder
Microphone
analog
audio
Analog
patchbay
analog
audio
digital
audio
analog audio
MIDI controllable analog mixer
ADC
digital
audio
analog audio
Loudspeakers
Amplifier
analog audio
DAC
DAT

Figura 3

UN SISTEMA PER LA SINTESI DI PARTITURE MIDI MEDIANTE ESECUZIONE DI MODELLI FORMALI°

G. Haus, A. Sametti

L.I.M. - Laboratorio di Informatica Musicale, Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano, via Moretto da Brescia 9, I-20133 Milano, Italia
E-mail: music@imiucca.unimi.it
Tel: +39 2 7575249 Fax: +39 2 76110556

Parole chiave: oggetti musicali, operatori musicali, algoritmi musicali, reti di Petri, modelli formali, temporizzazione, concorrenza, sincronizzazione, gerarchia, macrodefinizioni, sintesi di partiture, interfaccia utente iconica.

ABSTRACT

In this paper we show how we can synthesize music scores by means of formal models execution. The kind of models we use is an "ad hoc" arrangement of Petri Nets and a music algebra. While we associate music objects to *place* nodes for describing music information, we associate causal relationships and transformation rules to PN *structures* and PN *parameters* (marking, numeric labels, algorithms, etc.). music objects may be processed modifying the superimposition and juxtaposition laws within PN structure; on the other hand, PN parameters allow us to create *instances* of music objects which modify themselves according to the behaviour of PN models during execution. Music objects are coded according to three syntaxes: a coding language which is specific of ScoreSynth, an alphanumeric MIDI coding language and a coding format which implements the Standard MIDI Files 1.0 format. Music objects may be defined either by a common text editor or a MIDI controller (a keyboard, a guitar, a microphone, etc.). Music transformations are described by expressions associated to *transitions* and are coded by operators of the music algebra we have defined. Operators apply on pitches, durations, dynamics and ordering. Our approach makes possible the description of music objects and their transformations within compositions unbinded from the symbolic level of representation which is typical of common music notation. So, music appear as a multilayered model in which the musician chooses the number and the qualities of the various levels of representation. The ScoreSynth system has two user environments: the Editor and the Executer of the formal models. The Editor allows the user to describe hierarchical, timed, concurrent, deterministic/non-deterministic and weighted Petri Net models; both places and transitions are used as the "morphism" nodes to implement hierarchy. The Executer executes the models, synthesizes MIDI scores and shows a graphic map of the music objects as they are generated by the running model; the musician, while looking at the graphic map, can interact with the Executer to proceed step by step and to know about the music objects which are generated and their relationships with other objects and music functions of the model. Both the Petri Net models (in the iconic form of representation) and the graphic maps of music objects can be printed. The Editor is able to process common Petri Net structures as macros; in this way, it is possible to call a Petri Net macro with a list of parameters to obtain an actual Petri Net structure within the growing model. A library of the most common macro nets is provided with the system.

° Questa ricerca è parzialmente supportata dal Consiglio Nazionale delle Ricerche (CNR) nell'ambito della LRC C4: MUSIC (Contratti N° 89.00031.69 e 90.00678.PF69), "STAZIONE DI LAVORO MUSICALE INTELLIGENTE", Progetto C: SISTEMI AVANZATI DI PRODUTTIVITA' INDIVIDUALE, Sottoprogetto 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, Progetto Finalizzato SISTEMI INFORMATICI E CALCOLO PARALLELO.

1. Introduzione

In questo articolo, è nostra intenzione mostrare come sia possibile descrivere, elaborare e sintetizzare strutture musicali tramite un tipo di rappresentazione più flessibile rispetto al tradizionale pentagramma; infatti, nella notazione usuale i simboli sono scelti e organizzati in base alle necessità strumentali; il livello di rappresentazione all'interno delle partiture è molto più dettagliato di quello della composizione musicale, e più astratto rispetto a quello relativo alla modellazione del timbro; il nuovo tipo di rappresentazione da noi proposto, crea una struttura musicale concettuale, con tanti livelli di astrazione quanti il compositore ne richiede e ci permette di descrivere ed elaborare esplicitamente ciò che noi chiamiamo *oggetti musicali* (siano essi oggetti musicali tradizionali o meno). L'oggetto musicale rappresenta una qualsivoglia porzione di informazione che possa avere un significato in musica, e che da noi è considerato un'entità, sia essa semplice o complessa, astratta o concreta, un'entità con un nome e relazioni con altri oggetti musicali. Possiamo descrivere oggetti musicali attraverso vari livelli di astrazione, all'interno di un contesto descrittivo gerarchico; ad esempio, possiamo avere i seguenti: il livello strutturale, il livello di partitura, il livello del timbro.

La notazione musicale tradizionale è caratterizzata da molti differenti linguaggi (uno o più linguaggi per ogni livello di rappresentazione). La nostra ricerca mira a definire uno e un solo linguaggio, adattabile ad ogni livello di rappresentazione.

Per definire lo strumento di descrizione più appropriato, abbiamo cercato uno strumento formale che possedesse le seguenti caratteristiche:
* quantità limitata di simboli;
* si rappresenta graficamente;
* permette la descrizione gerarchica;
* permette la descrizione dell'elaborazione degli oggetti musicali;
* permette la descrizione del tempo;
* permette modelli deterministici e non deterministici:
* permette macro definizioni per le strutture comuni;
* mostra graficamente i passaggi della sintesi della partitura durante l'esecuzione del modello formale.

2. Modellazione di partiture mediante reti di Petri e un'algebra musicale

Usiamo le reti di Petri (Degli Antoni & Haus, 1983) da oltre dieci anni, come strumento base per la descrizione e l'elaborazione dell'informazione musicale. Per confronto si può considerare anche l'approccio affine seguito in (Pope, 1986). Durante la nostra ricerca, abbiamo sviluppato alcuni programmi per scrivere ed eseguire reti di Petri sia in ambito musicale, sia in applicazioni "general purpose" (Haus & Rodriguez, 1989). Il più recente programma da noi sviluppato è ScoreSynth (Haus & Sametti, 1991), nel quale usiamo reti di Petri per la descrizione, l'elaborazione e la sintesi delle partiture musicali; mentre associamo oggetti musicali a nodi di tipo posto che descrivono l'informazione musicale, associamo relazioni causali e regole di trasformazione alle strutture ed ai parametri delle reti di Petri (marcatura, etichette numeriche, algoritmi, etc.). Gli oggetti musicali possono essere elaborati modificando le leggi di sovrapposizione e sequenzialità all'interno della struttura di rete; contemporaneamente, i parametri delle reti di Petri ci permettono di creare istanze di oggetti musicali che si modificano in base al comportamento della rete di Petri stessa, durante l'esecuzione. Inoltre, è possibile rappresentare la medesima informazione musicale tramite reti di Petri ad un livello di astrazione più alto o più basso, attraverso vari approcci di modellazione (Bertoni et al., 1978; Goguen, 1975).

Una rete di Petri viene generalmente definita da una quadrupla **<P, T, A, M>**, dove **P** è l'insieme dei posti, **T** è l'insieme delle transizioni, **A** è l'insieme degli archi che connettono posti a transizioni e

transizioni a posti, **M** è la marcatura della rete. La nostra implementazione delle reti di Petri estende la classica definizione data dallo stesso Petri in (Petri, 1976), ben esemplificata rispetto alle proprietà descrittive delle reti nella definizione di modelli formali in (Peterson, 1981), in modo da permettere al musicista la costruzione di modelli gerarchici costituiti fondamentalmente da reti, oggetti musicali e algoritmi musicali a loro volta costituiti da espressioni a operatori. Secondo questo approccio, una rete di Petri è completamente definita mediante la tripla **<P, T, A>** e le seguenti tre:

(1) **P ∋ p** ≡ *<identificatore, marche, capacità, canale MIDI, oggetto, esecuzione, file >*

(2) **T ∋ t** ≡ *<identificatore, algoritmo >*

(3) **A ∋ a** ≡ *<nodo-da, nodo-a, molteplicità >*

dove:

P è l'insieme dei posti; **T** è l'insieme delle transizioni; **A** è l'insieme degli archi; *identificatore* è l'etichetta del nodo (sia esso un posto o una transizione); *marche* è il numero di marche contenute nel posto cioè il valore della marcatura **M** (vedi più avanti); *capacità* è il massimo numero di marche ammesso per quel posto; *canale MIDI* è il numero di canale MIDI a cui l'oggetto musicale associato a quel posto, se esiste, deve essere inviato per l'esecuzione; *oggetto* definisce se c'è un oggetto musicale associato a quel posto; *esecuzione* definisce se l'oggetto musicale associato a quel posto, se esiste, deve essere eseguito; *file* definisce se l'oggetto musicale associato a quel posto, se esiste, è memorizzato su file; *algoritmo* definisce se l'algoritmo musicale associato a quella transizione, se esiste, deve essere eseguito; *nodo-da* è l'identificatore del nodo da cui parte l'arco; *nodo-a* è l'identificatore del nodo in cui termina l'arco; *molteplicità* è l'etichetta numerica dell'arco. Se un posto ha un oggetto musicale associato e memorizzato su file, il suo identificatore rappresenta sia l'etichetta del nodo che l'identificatore del file corrispondente.

P, **T** e **A** sono insiemi finiti; **P** non può essere vuoto. Sia oggetti che algoritmi musicali hanno le loro definizioni sintattiche (vedi § 3. e § 4. rispettivamente).

La marcatura di una rete è la distribuzione di marche nei nodi di tipo posto; il numero di marche e la loro distribuzione varia durante l'esecuzione della rete. La marcatura è definita dal seguente vettore:

(4) $\mathbf{M} \equiv (\mathbf{m_1}, \mathbf{m_2}, \ldots, \mathbf{m_i}, \ldots, \mathbf{m_n})$

dove **n** è il numero di posti della rete, **N** è l'insieme dei numeri interi positivi e

(5) per ogni $\mathbf{N} \ni \mathbf{m_i} = \mathbf{M}(\mathbf{p_i})$ con $\mathbf{P} \ni \mathbf{p_i}$.

Inoltre, la gerarchia del modello è descritta dall'insieme:

(6) **S ∋ s** ≡ *<identificatore, nodo-inizio, nodo-fine, livello-di-ricorsione >*

costituito dall'insieme delle connessioni gerarchiche, dove:

identificatore è l'etichetta della rete, *nodo-inizio* è l'identificatore del nodo di inizio dell'applicazione del morfismo, *nodo-fine* è l'identificatore del nodo di fine dell'applicazione del morfismo, *livello-di-ricorsione* è il livello di ricorsione di quella rete. Discuteremo la nostra implementazione di gerarchia e di ricorsione nel § 6.

L'insieme **S** rappresenta tutta l'informazione attinente la gerarchia del modello mentre ogni tripla di insiemi **P**, **T**, **A** descrive completamente una particolare rete del modello; in altre parole, un modello è completamente definito quando sono dati un insieme **S** e tante triple **<P, T, A>** quante sono le reti del modello.

Una considerazione generale: un modello di reti di Petri con un particolare stato iniziale (cioè la marcatura iniziale) può rappresentare una famiglia di partiture; una esecuzione particolare del modello effettua la sintesi di una specifica partitura. Cambiando la marcatura iniziale del modello, modifichiamo

la famiglia di partiture che possono essere generate attraverso l'esecuzione del modello. Un'eccezione: quando utilizziamo un modello pienamente deterministico, possiamo produrre un'unica partitura (data una particolare marcatura iniziale).

Qui di seguito descriviamo per sommi capi l'algoritmo di esecuzione delle reti dei modelli formali implementato in ScoreSynth:

```
procedure Esegui;
begin
   repeat
      Blocco := NOT ( Scatto ); {vd. funzione Scatto di seguito }
      if ... nessun processo musicale attivo... then Nessun_Posto_Attivo := TRUE
      else
         begin
            Nessun_Posto_Attivo := FALSE;
            calcola il DeltaTime per i processi (uno o più) che terminano per primi;
            esegui tutti i processi attivi per DeltaTime unità di tempo;
            elimina tutti i processi terminati dalla lista dei processi attivi;
            incrementa di DeltaTime unità la variabile che simula lo scorrere del tempo;
         end;
   until Blocco & Nessun_Posto_Attivo ;
end;

function Scatto: boolean;{Restituisce TRUE se almeno una transizione è scattata}
begin
   repeat
      crea la lista delle transizioni che possono scattare "semplicemente";
      crea la lista delle transizioni abilitate allo scatto nel contesto di situazioni di alternativa e/o conflitto;
      if ... se esiste almeno una transizione che può scattare ... then
         begin
            Scatto:= TRUE;
            while ...se esiste almeno una transizione in alternativa e/o conflitto ... do
               begin
                     genera un indice pseudocasuale con distribuzione uniforme per
                     le transizioni che possono scattare in situazioni di alternativa e/o conflitto;
                     esegui la transizione indicata dal numero generato;
                     elimina dalla lista tutte le transizioni che non sono più in alternativa e/o conflitto;
               end;
            esegui tutte le transizioni che possono scattare "semplicemente";
         end
      else
         Scatto:= FALSE;
   until ...non esistono più transizioni abilitate ... ;
end;
```

Per completezza, descriviamo brevemente anche la procedura che esegue lo scatto delle transizioni:

```
procedure EseguiTransizione(TransId);
         {TransId identifica la transizione da eseguire}
begin
   if ...la transizione ha un algoritmo associato ... then esegui l'algoritmo;
```

modifica la marcatura dei posti in ingresso secondo le regole dello scatto e la molteplicità degli archi;
modifica la marcatura dei posti in uscita secondo le regole dello scatto e la molteplicità degli archi;
per tutti i posti in uscita:
if *...il posto ha un oggetto musicale associato ...* **then**
inserisci l'oggetto musicale nella lista dei processi attivi;
end;

Il sistema di ScoreSynth è costituito da un Editor e da un Esecutore dei modelli. L'Editor permette la descrizione gerarchica, temporizzata, concorrente, deterministica/non deterministica e pesata dei modelli generati tramite reti di Petri; sia i posti sia le transizioni sono usati come morfismi per definire la gerarchia. Strutture comuni di reti di Petri possono essere descritte come macro. Un'interfaccia utente grafica ed interattiva permette di rappresentare reti di Petri per mezzo di icone (vedi Fig. 1) e di descrivere i vari attributi dei nodi e gli oggetti ed algoritmi ad essi associati mediante specifici dialoghi a finestra.

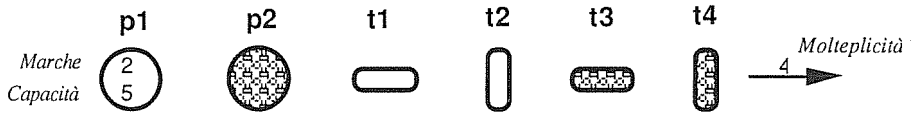


Figura 1: icone utilizzate dall'editor di reti di ScoreSynth; da sinistra: nodo di tipo posto con marche e capacità, posto che rappresenta una rete di livello inferiore, nodo di tipo transizione, transizione posizionata ortogonalmente, transizione che rappresenta una rete di livello inferiore, transizione che rappresenta una rete di livello inferiore posizionata ortogonalmente, arco orientato con molteplicità.

L'Esecutore esegue i modelli, sintetizza partiture MIDI e traccia una mappa grafica degli oggetti musicali, mentre essi sono generati durante l'esecuzione; il musicista, seguendo la mappa grafica, può interagire con il modello. Il musicista può arrestare l'esecuzione in caso di errori del modello ed entrare nell'ambiente Editor per modificarlo. Tutto ciò è reso possibile grazie all'esecuzione del modello e alla finestra grafica che mostra la mappa degli oggetti musicali su un piano tempo / canali MIDI nel momento in cui essi sono inseriti nella partitura, oltre ad ogni altra informazione richiesta dal musicista, relativa alla creazione degli oggetti musicali (vedi Fig. 2).

I paragrafi seguenti spiegano le caratteristiche del sistema ScoreSynth. ScoreSynth è stato sviluppato su computer Apple Macintosh®, in ambiente MPW.

3. Oggetti musicali

Il significato di oggetto musicale, già citato, è strettamente legato ai nodi di tipo posto. Con oggetto musicale indichiamo non solo le sequenze di note, ma anche un qualcosa di più generale, non esclusivamente collegato all'ascolto o all'idea di processo; per quanto riguarda la misura del tempo, esso potrebbe durare zero secondi, ad esempio nel caso di un oggetto costituito solo da comandi di controllo per uno strumento MIDI per la sintesi del suono. Naturalmente, la definizione di oggetti musicali è parzialmente limitata dagli standard di codifica dell'informazione musicale che usiamo (MIDI) e non deve uscire dagli standard stessi. Tutti i messaggi MIDI, anche se non strettamente legati ad una nota, possono essere inseriti in un oggetto musicale.

Gli oggetti musicali possono essere codificati secondo tre sintassi: un linguaggio di codifica, specifico di ScoreSynth, un linguaggio alfanumerico di codifica MIDI e un formato di codifica che implementa gli Standard MIDI Files 1.0 (sia tipo 0 sia tipo 1). Il primo formato è usato all'interno del programma. Il secondo è stato scelto per la presenza di strumenti di editing. Il terzo formato è stato implementato per l'importazione/esportazione dei file. Gli oggetti musicali possono essere definiti sia da un comune editor di testo sia da un controller MIDI (una tastiera, una chitarra, un microfono, etc.). Essi possono essere memorizzati in file separati (un file per ogni oggetto) o all'interno del file della rete al quale il posto cui è associato appartiene.

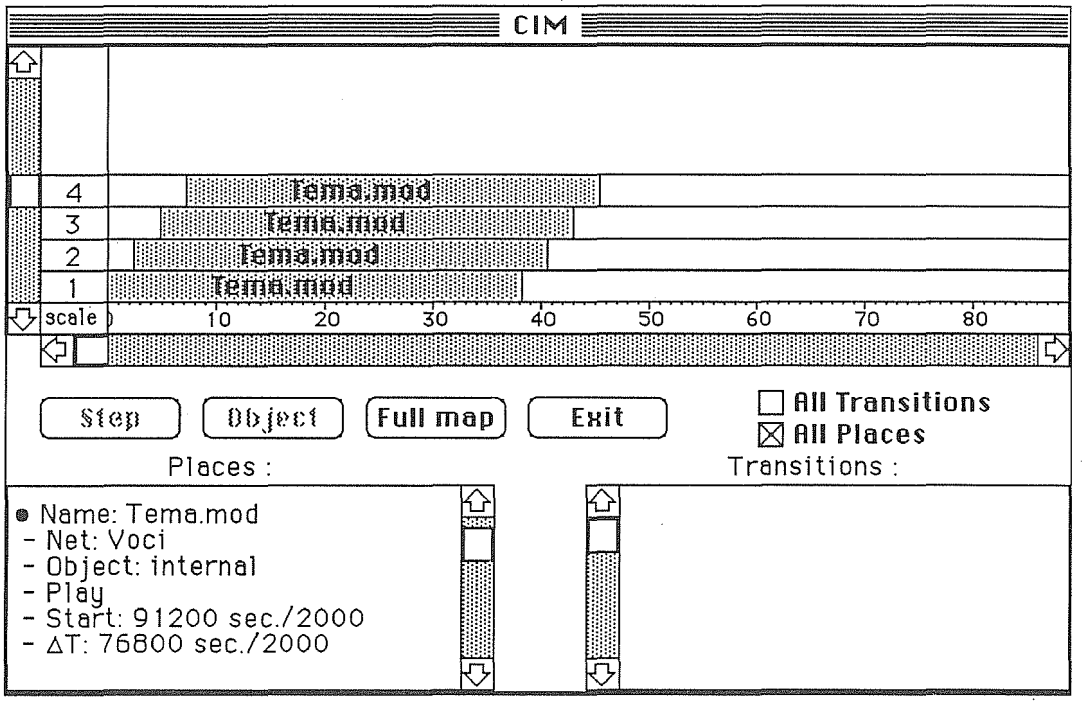


Figura 2: mappa di dialogo degli oggetti musicali con le due finestre relative agli attributi dei nodi della Rete.

Diamo qui di seguito la definizione sintattica di oggetto musicale in forma di BNF:

< MusicObject > ::= < MusicObject > . < MetaEvent > |
< MusicObject > < MIDIEvent > |
. < MetaEvent > |
< MIDIEvent >

< MetaEvent >	::=	***Tempo*** < integer > |
		Base < integer > |
		Signature < integer > < integer > |
		Rest < integer > |
		MidiEx < MIDIExBytes >
< MIDIExBytes >	::=	" *...una qualunque sequenza di byte, in esadecimale, in accordo con la specificazione dello standard MIDI ...*"
< MIDIEvent >	::=	< Header > ***Pgm*** < integer > |
		< Header > ***After*** < integer > |
		< Header > ***Polyp*** < integer > |
		< Header > ***Ctrl*** < integer > < integer > |
		< Header > ***Bend*** < integer > |
		< Header > < NoteEvent >
< Header >	::=	< Channel > < Quarter > < Offset >
< NoteEvent >	::=	< Duration > < Pitch > < KeyVel >
< Pitch >	::=	< Note+Mod > < Octave > | < KeyNumber >
< Note+Mod >	::=	< Note > | < Note > #
< Note >	::=	***A | B | C | D | E | F | G***
< Channel >	::=	***1*** | ... | ***16***
< Quarter >	::=	< positive integer > *{...dipende dalla lunghezza del brano}*
< Offset >	::=	< positive integer > *{...dipende da* ***.Base}***
< Duration >	::=	< positive integer >
< KeyVel >	::=	***0*** | ... | ***127***
< Octave >	::=	***-2*** | ... | ***8***
< KeyNumber >	::=	***0*** | ... | ***127***

La più semplice rete per eseguire un oggetto musicale è mostrata in Fig. 4a. Abbiamo unito al posto **Tema** l'oggetto musicale **Tema** che era stato in precedenza memorizzato in un file (vedi Fig. 3), secondo le specifiche di un qualunque formato dei tre accettabili da ScoreSynth. L'attributo superiore del posto stabilisce il numero di marche disponibili, mentre quello inferiore indica la massima capacità di marche permesse a quel posto.

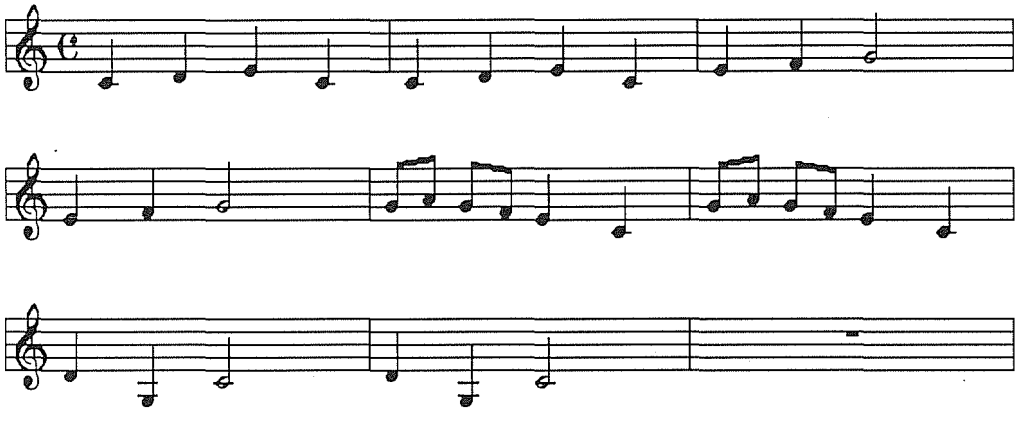


Figura 3: la partitura del tema di "Fra Martino" salvata nel file **Tema**.

4. L'algebra musicale

L'idea principale, è quella di poter trasformare attraverso algoritmi sia i parametri dei suoni (altezza, timbro, intensità, durata) sia l'ordine in cui le note appaiono. Gli algoritmi musicali sono associati alle transizioni. Se una transizione ha un algoritmo associato, ScoreSynth fonde tutti gli oggetti musicali in entrata, prima di effettuare lo scatto della transizione; successivamente, quando la transizione è scattata, esso toglie una marca da ogni posto di entrata, effettua la trasformazione sul singolo oggetto musicale ottenuto, una marca viene aggiunta a tutti i posti in uscita e l'oggetto musicale viene associatto a ogni posto in uscita permesso. La trasformazione è applicabile solo agli eventi di tipo nota. Tutti gli altri comandi MIDI sono filtrati ed eliminati.

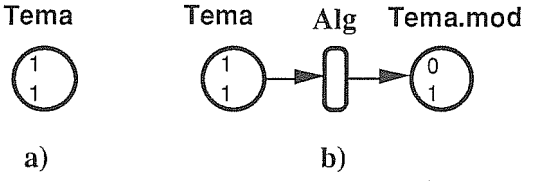


Figura 4: a) un posto associato ad un oggetto musicale;
b) una semplice rete con un algoritmo associato alla transizione **Alg**.

Se **Tema** rappresenta l'oggetto musicale sul quale l'algoritmo è eseguito in entrata, l'algoritmo può essere applicato all'intero tema oppure solo ad una sua sottosequenza, scelta dal musicista. E' inoltre possibile utilizzare le informazioni di ogni oggetto musicale disponibile in quel momento nell'ambito dell'intero modello. Un algoritmo può essere costituito da uno o più singoli operatori. Ogni operatore può modificare i parametri delle note oppure il loro ordine. Ogni operatore è applicato all'oggetto musicale risultante dall'operatore precedente. L'algoritmo è applicato alla sequenza di note definita nella sua intestazione, via via per ogni nota. L'intestazione definisce inoltre il tipo di parametro al quale l'espressione dell'operatore deve essere applicata: altezza, durata, intensità (velocità dei tasti), canale MIDI e ordine delle note interessate. Diamo ora uno sguardo a due esempi di tali algoritmi.

Inversione speculare. Può essere eseguita attraverso una nota di riferimento, chiamata "nota specchio": ogni valore di altezza che viene eseguito è simmetrico alla nota specchio. L'operatore seguente inverte le altezze rispetto al C#3: **P: 1, $, [Tema, 1], 2 * C#3 - ?** dove **P** rappresenta l'altezza; **1, $** significa: dalla prima all'ultima nota dell'oggetto musicale **Tema**; **[Tema, 1]** determina l'utilizzo delle informazioni dall'oggetto musicale **Tema** (non dagli oggetti musicali collegati ai posti di ingresso che sono fissi e non richiedono esplicita specificazione) a partire dalla prima nota (**1**); e **2 * C#3 - ?** è l'espressione che calcola i nuovi valori dell'altezza. **?** è un metacarattere che rappresenta il valore del parametro (l'altezza, in questo caso), riferito alla nota che è in corso di esecuzione.

Altri operatori che modificano i parametri delle note sono:

C (Channel) = operatore su canali MIDI;
D (Duration) = operatore sulla durata
L (Loudness) = operatore sull'intensità (velocità dei tasti).

Inversione dell'ordine (Retrogradazione). L'operatore di inversione ridefinisce l'ordine degli oggetti musicali in entrata: **I: 1, $** dove I significa inversione e **1, $** significa: dalla prima all'ultima nota.

Altri operatori che elaborano l'ordine delle note sono:

K (Kill) = operatore per cancellare sottosequenze di note;
S (Save) = operatore per conservare sottosequenze di note;
M (Multiply) = operatore per la moltiplicazione (ripetizione) di sottosequenze di note;
R (Rotate) = operatore per la rotazione di sottosequenze di note.

Una rete molto semplice per trasformare un oggetto musicale è riportata in Fig. 4b. E' possibile: aggiungere qualsiasi algoritmo vogliamo alla transizione **Alg**, scegliere se disabilitare l'esecuzione degli oggetti musicali **Tema** e/o **Tema.mod** (stabilendo gli attributi specifici del posto) e decidere se l'oggetto musicale ottenuto dopo la trasformazione (**Tema .mod**) sarà di tipo volatile o salvato in un file. In questo ultimo caso un algoritmo di inversione speculare potrebbe essere: **P: 1, $, 2 * C#3 - ?**

Diamo qui di seguito la definizione sintattica di algoritmo musicale in forma di BNF:

< Algorithm > ::= < Algorithm > < Operator > | < Operator >
< Operator > ::= < QualOp1 > *:* < RangeExpression > , < RangeExpression > |
< QualOp2 > *:* < RangeExpression > , < RangeExpression > ,
< OpExpression > |
< QualOp3 > < TonalExpression > *:* < RangeExpression >,
< RangeExpression > , < OpExpression > |
< QualOp3 > < TonalExpression > *:* < RangeExpression >,
< RangeExpression > , *[* < MusicObjectId > , < RangeExpression > *]*,
< OpExpression > |
< QualOp4 > *:* < RangeExpression > , < RangeExpression > ,
< OpExpression > |
< QualOp4 > *:* < RangeExpression > , < RangeExpression > ,
[< MusicObjectId > , < RangeExpression > *]* , < OpExpression >
< QualOp1 > ::= ***I*** | ***K*** | ***S***
< QualOp2 > ::= ***M*** | ***R***
< QualOp3 > ::= ***P***
< QualOp4 > ::= ***C*** | ***D*** | ***L***
<MusicObjectId > ::= "... *alphanumeric string* ..."
< TonalExpression > ::= *[* < Note > < Tonality > *]* | *[* < Note > *]*
< Tonality > ::= ***m*** | **-** | ***t*** | ***p***
< RangeExpression > ::= < RangeExpression > + < RangeTerm > |
< RangeExpression > - < RangeTerm > |
< RangeTerm >
< OpExpression > ::= < OpExpression > + < OpTerm > |
< OpExpression > - < OpTerm > |
< OpTerm >
< RangeTerm > ::= < RangeTerm > * < RangeFactor > |
< RangeTerm > / < RangeFactor > |
< RangeFactor >
< OpTerm > ::= < OpTerm > * < OpFactor > |
< OpTerm > / < OpFactor > |
< OpFactor >
< RangeFactor > ::= < integer > |
< MetaChar >|
(< RangeExpression >)

< OpFactor >	::=	< Pitch > |
		< integer > |
		< MetaChar >|
		(< OpExpression >)
< MetaChar >	::=	*$ | ! | % | ?*
< Pitch >	::=	< Note+Mod > < Octave > | < KeyNumber >
< Note+Mod >	::=	< Note > | < Note > #
< Note >	::=	*A | B | C | D | E | F | G*
< Octave >	::=	*-2 | ... | 8*
< KeyNumber >	::=	*0 | ... | 127*

5. Scatto delle transizioni e temporizzazione

Le reti di Petri si prestano particolarmente per la descrizione ed esecuzione di processi concorrenti e per il controllo della loro sincronizzazione. E' il comportamento della rete che determina implicitamente la temporizzazione. Possiamo sincronizzare oggetti musicali semplicemente tramite un'opportuna struttura dei collegamenti tra i nodi. E' la struttura della rete che determina la sequenza degli scatti.

In generale (una rete contenente costrutti non deterministici), la sequenza degli scatti delle transizioni cambia sulla base delle differenti esecuzioni della rete; in questo modo non c'è corrispondenza tra una rete e la sua sequenza di scatti. Possiamo avere molte transizioni abilitate per scattare nello stesso momento e non possiamo sapere quale scatterà per prima: la rete non fornisce questo tipo di informazione. Ogni esecuzione di una rete di Petri può fornire una sequenza di scatti diversa. Il nostro approccio rappresenta oggetti musicali temporizzati in sé e le loro trasformazioni determinate dalle transizioni. Lo scatto di una transizione ha durata nulla. In questo modo, quando una marca è messa in un posto a cui è associato un oggetto musicale, la marca non può essere considerata per lo scatto delle transizioni collegate al posto fino a quando l'oggetto musicale associato non sia terminato.

6. Gerarchia e macro

Introduciamo ora due concetti: gerarchia (raffinamento) e macro. Un raffinamento, chiamato sottorete, è una rete di Petri che fornisce una descrizione più dettagliata del nodo (sia esso un posto o una transizione) del livello di astrazione immediatamente superiore (Kotov, 1978). Se scegliamo di definire una sottorete associata ad un posto **P**, la rete "figlia" deve avere due posti speciali: un posto di entrata **IN** e un posto di uscita **OUT**; gli archi di entrata di **P** diventano gli archi di entrata di **IN** e gli archi di uscita di **P** diventano archi di uscita di **OUT**. Le transizioni possono essere raffinate in modo del tutto analogo. Possiamo anche modellare reti ricorsive, reti che contengono se stesse come nodi di raffinamento; per evitare espansioni ricorsive all'infinito, dobbiamo assegnare ad ogni rete ricorsiva il suo livello di ricorrenza (vedi definizione dell'insieme **S** al § 2.).

Inoltre, durante la stesura di un modello di reti, dobbiamo spesso creare molte reti simili, che differiscono nei loro attributi dei posti e negli algoritmi, ma che sono identici nella struttura. Con una macro rete, possiamo usare un'unica rete come modello base e quindi modificare i suoi attributi e/o algoritmi secondo le nostre necessità. Quindi il musicista ha anche la possibilità di crearsi delle librerie personali, che contengono le reti da lui più comunemente utilizzate. Ciò è reso possibile grazie alla definizione, per ogni posto della sottorete o transizione della sottorete (a nostra discrezione), di una lista di modificatori applicabili alla rete; tale lista è disponibile in memoria e viene attualizzata prima dell'esecuzione del modello.

Diamo qui di seguito la definizione sintattica di lista di modificatori in forma di BNF:

< ModifierList >	::=	< ModifierList > < Modifier> | <Modifier>
< Modifier>	::=	< PlaceModifier > | < TransModifier > | < ArcModifier >
< PlaceModifier >	::=	< PlaceId > *:* < PlaceOpList >
< PlaceOpList >	::=	< PlaceOpList > < PlaceOp > | < Place Op >
< PlaceOp >	::=	< PlaceAttribute > | < PlaceSetting >
< PlaceAttribute >	::=	***play*** | ***mute*** | ***file*** | ***notfile*** | ***obj*** | ***notobj*** | ***simple*** | ***subnet***
< PlaceSetting >	::=	< NewId > |
		C = < Capacity > |
		M = < Token > |
		CH = < Channel > |
		{ < MusicObject > *}*
< TransModifier >	::=	< TransId > *:* < TransOpList >
< TransOpList >	::=	< TransOpList > < TransOp > | < TransOp >
< TransOp >	::=	< TransAttribute > | < TransSetting >
< TransAttribute >	::=	***enabled*** | ***disabled***
< TransSetting >	::=	< NewId > |
		{ < Algorithm > *}*
< ArcModifier >	::=	< PTArcOp > | < TPArcOp >
< PTArcOp >	::=	< PlaceId > -> < TransId > = < Multiplicity >
< TPArcOp >	::=	< TransId > -> < PlaceId > = < Multiplicity >
< Multiplicity >	::=	***1*** | ... | ***240***

Ora forniamo un esempio a titolo di ulteriore chiarimento. Le Figg. 5a, 5b e 6a rappresentano un modello deterministico basato sul tema di "Fra Martino" mentre le Figg. 5a, 5b e 6b forniscono una versione non deterministica. Può essere interessante confrontare il nostro modello con quelli sviluppati in (Greussay, 1973; Smoliar, 1971). La rete più astratta è la **CIM**. Il posto **Fra Martino** è dettagliato dalla rete omonima dove **Counter** è il posto di entrata e **Voci** è il posto di uscita. L'arco di entrata del posto **Counter** ha peso 3: ciò significa che lo scatto della transizione metterà tre marche nel posto **Counter** (Valk, 1978). Il posto **Pausa 4/4** ha associato un oggetto musicale corrispondente a una pausa di 4/4. Per espandere il posto **Voci** con una macro rete possiamo utilizzare una delle due reti della Fig. 6. Nel primo caso otteniamo un'esecuzione deterministica del modello; nel secondo una esecuzione non deterministica: ciò significa che non possiamo fissare la sequenza degli scatti, ma essa è determinata dall'algoritmo di esecuzione del modello di ScoreSynth (vedi § 2.) secondo una distribuzione pseudocasuale e uniforme dell'ordine. Il posto di entrata non ha marca iniziale perchè riceverà marche dal livello superiore del modello (sia nelle rete **Fra Martino** che in **Voci**). La lista dei parametri che permette il richiamo della macro rete **Voices** è la seguente (si noti che richiamiamo solo ora l'attualizzazione degli algoritmi associati alle quattro transizioni della macro rete, ma possiamo agire nello stesso modo, rispettando tutti i possibili attributi della rete di Petri):

Alg1:	{C: 1, $, [Tema, 1], 1	{...tutti su canale MIDI 1; usa oggetto musicale "Tema"}
	M:1, $, 2}	{...ripeti 2 volte}
Alg2:	{C: 1, $, [Tema, 1], 2	{...tutti su canale MIDI 2; usa oggetto musicale "Tema"}
	M:1, $, 2	{...ripeti 2 volte}
	P[C]: 1, $, ? + 2}	{...trasposizione di 2 gradi, tonalità *do maggiore* }
Alg3:	{C: 1, $, [Tema, 1], 3	{...tutti su canale MIDI 3; usa oggetto musicale "Tema"}
	D: 1, $, ? / 2	{...dimezza le durate}
	M:1, $, 4}	{...ripeti 4 volte}
Alg4:	{C: 1, $, [Tema, 1], 4	{...tutti su canale MIDI 4; usa oggetto musicale "Tema"}
	I: 1, $	{...retrogrado}
	M:1, $, 2}	{...ripeti 2 volte}

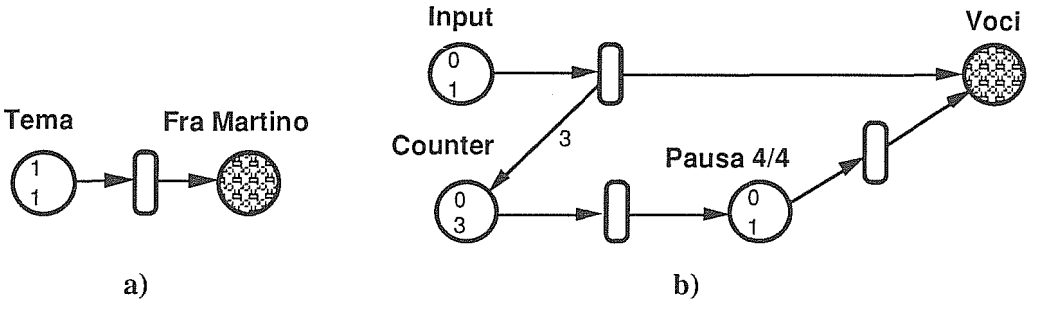


Figura 4: a) la rete **CIM**;
b) la rete **Fra Martino**.

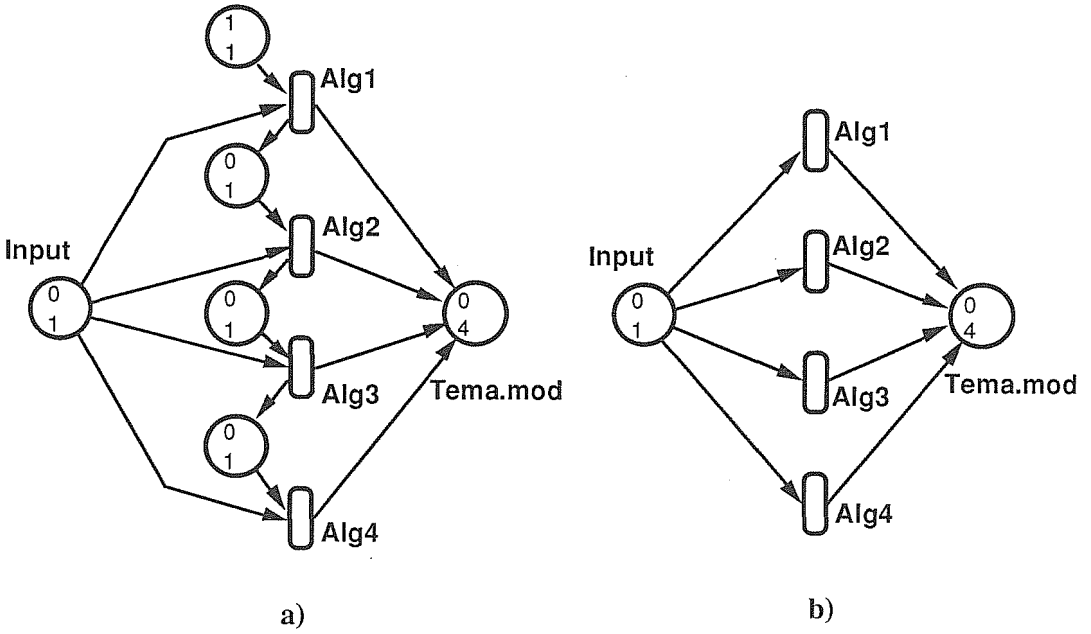


Figura 6: a) versione deterministica della rete **Voices**;
b) versione non deterministica della rete **Voices**.

La Fig. 2 mostra la mappa grafica degli oggetti musicali ottenuti dall'esecuzione del modello **CIM**; come possiamo vedere, ci sono quattro differenti istanze dell'oggetto musicale **Tema.mod** che appartengono alla rete **Voices**; essi sono messi su differenti canali MIDI. Sono oggetti musicali differenti tra loro perchè generati attraverso differenti algoritmi. Gli altri oggetti musicali del modello non

sono rappresentati nella mappa grafica (e quindi nella partitura di uscita) perchè: **Tema** è stato definito come oggetto musicale che non deve essere eseguito e **Rest 4/4** produce un ritardo essendo esso un oggetto musicale di tipo pausa, ma non corrisponde ad un evento MIDI.

7. Conclusioni

La nostra ricerca ha dimostrato che le strutture musicali possono essere trasformate semplicemente:
* modificando la marcatura, le specifiche del canale MIDI, le capacità dei posti;
* modificando le etichette degli archi;
* modificando oggetti musicali associati a posti;
* modificando algoritmi associati a transizioni;
* modificando le strutture delle reti di Petri;
* eseguendo più volte modelli non deterministici di reti di Petri.

Mentre la costituzione delle strutture di reti di Petri e i parametri modificano le relazioni causali/strutturali all'interno dell'architettura delle partiture, l'editing di oggetti musicali e algoritmi modifica le unità di informazione di base e le loro trasformazioni. Il sistema ScoreSynth risulta essere un potente strumento in grado di descrivere e eseguire musica, vicino al pensiero e alla percezione musicale più della comune notazione musicale. Una raccolta di composizioni musicali realizzate mediante reti di Petri come strumento di descrizione è pubblicata in (Haus, 1990).

Ringraziamenti

Ringraziamo E. Bianchi, A. Camurri, G. Degli Antoni, G. Jacomini, A. Rodriguez e R. Zaccaria per i loro contributi alle ricerche e esperienze precedenti sulle reti di Petri e la musica. Un ringraziamento particolare a S. Scolaro per il suo determinante contributo alla definizione e allo sviluppo di alcune caratteristiche di ScoreSynth Rel. 2.0 (in particolare l'interfaccia grafica e le reti di Petri macro). Inoltre, un grazie speciale allo staff del L.I.M. - Laboratorio di Informatica Musicale, per il supporto scientifico e tecnico costantemente fornito durante tutta la nostra ricerca.

Riferimenti bibliografici

A. Bertoni, G. Haus, G. Mauri, M. Torelli, "A Mathematical Model for Analyzing and Structuring Musical Texts", **Interface**, Vol. 7, N.1, Swets & Zeitlinger B.V., Amsterdam, 1978, pp.31-43.

G. Degli Antoni, G. Haus, "Music and Causality", **Proc. 1982 ICMC**, Venezia, Computer Music Association Ed., San Francisco, 1983, pp. 279-296.

J. A. Goguen, "Complexity of Hierarchically Organized Systems and the Structure of Musical Experience", **UCLA Computer Science Dept. Quarterly**, Vol. 3, N. 4, 1975.

P. Greussay, **Modeles de descriptions symboliques en analyse musicale**, M. Th. in Linguistics and Informatics, Université Paris VIII, Paris, 1973.

G. Haus, **Musiche per poche parti**, Stile Libero/Virgin Dischi, SLCD1012, Milano, 1990.

G. Haus, A. Rodriguez, "Music Description and Processing by Petri Nets", **1988 Adv. Petri Nets**, LNCS, N.340, Springer, Berlin, 1989, pp.175-199.

G. Haus, A. Sametti, "SCORESYNTH: a System for the Synthesis of Music Scores based on Petri Nets and a Music Algebra", **IEEE Computer**, Vol. xxx, N. 7, 1991.

V. E. Kotov, "An Algebra for Parallelism based on Petri Nets", **MFCS 1978**, Proc. 7th Symposium, Zakopane, Polonia, Springer, Berlin, 1978.

J. L. Peterson, **Petri Net Theory and the Modeling of Systems**, Prentice Hall, New Jersey, 1981.

C. A. Petri, "General Net Theory", **Proc. Joint IBM & Newcastle upon Tyne Seminar on Computer Systems Design**, 1976.

S. Pope, "The Development of an Intelligent Composer's Assistant", **Proc. 1986 ICMC**, Den Haag, Computer Music Association Ed., San Francisco, 1986, 16 pp.

S. W. Smoliar, **A Parallel Processing Model of Musical Structures**, PhD. Th. MIT, TR-242, MIT, Cambridge, MA, 1971.

R. Valk, "Self-Modifying Nets, a Natural Extension of Petri Nets", **ICALP 1978**, LNCS, N. 62, Springer, Berlin, 1978, pp. 464-476.

UN AMBIENTE INTEGRATO PER LA ELABORAZIONE MUSICALE MIDI E LA SINTESI DEL SUONO MEDIANTE MODULI DSP°

A. Ballista, E. Casali, J. Chareyron, G. Haus

L.I.M. - Laboratorio di Informatica Musicale, Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano, via Moretto da Brescia 9, I-20133 Milano, Italia
E-mail: music@imiucca.unimi.it
Tel: +39 2 7575249 Fax: +39 2 76110556

Parole chiave: sequencing, MIDI, digital signal processing, microprogrammazione, programmazione timbrica, mixing

ABSTRACT

In this paper the Sound Processing Environment (SPE) of the Intelligent Music Workstation (IMW) is described. IMW-SPE integrates MIDI devices and DSP units allowing the user to control both MIDI devices and DSP units within the same sequencing environment so that DSP units may be considered if they were real time MIDI devices with timbre controlling capabilities. IMW-SPE consists of three subsystems: the MIDI/DSP sequencing program (SQAN) and two timbre programming software modules (Waver and LASy Workbench) which implement a number of digital sound synthesis techniques by DSP units. The end-user has complete control of the whole environment by means of MIDI programming only and without the need of direct DSP microprogramming.

The Waver Workbench allows sound synthesis by Two-Variable Functions, Waveshaping and analysis/synthesis by Wavelet Transforms. The LASy Workbench implements Linear Automata Synthesis. The SQAN DSP microprogram library includes Two-Variable Functions, Waveshaping and Linear Automata Synthesis sound synthesis units.

IMW-SPE is an open environment so that a musician can add his own sound synthesis units. It is also possible to import/export both sounds and MIDI files by means of Digidesign Sound Designer II and Standard MIDI File 1.0 formats respectively.

This work has been developed in the frame of the Intelligent Music Workstation (IMW) Project which is devoted to the design and development of a music software/hardware environment in which commercially available products are integrated with prototypal modules built up in the frame of musical informatics research.

° Questo lavoro è stato realizzato con il supporto e nell'ambito della LRC C4 MUSIC: "STAZIONE DI LAVORO MUSICALE INTELLIGENTE", obiettivo C: SISTEMI AVANZATI DI PRODUTTIVITA' INDIVIDUALE, sottoprogetto 7: SISTEMI DI SUPPORTO AL LAVORO INTELLETTUALE, progetto finalizzato SISTEMI INFORMATICI E CALCOLO PARALLELO del Consiglio Nazionale delle Ricerche (Contratti N° 89.00031.69 e 90.00678.PF69). Il metodo di controllo delle schede DSP come unità periferiche MIDI è stato oggetto di brevetto industriale da parte del Consiglio Nazionale delle Ricerche congiuntamente agli autori.

1. Introduzione

Negli ultimi dieci anni le applicazioni di Computer Music si sono prevalentemente sviluppate in due ambiti paralleli, il primo basato su dispositivi MIDI (sintetizzatori, campionatori, mixer, effetti, sequencer) e il secondo basato su unità DSP (add-on board, unità stand-alone, sistemi DSP ad architettura matriciale).

Grazie alla loro facilità d'uso, all'affidabilità, al basso costo ed alla disponibilità di software per la loro gestione, i dispositivi MIDI hanno avuto una grande diffusione mentre così non è stato per le unità DSP a causa della loro difficile programmazione, della mancanza di software dedicato e della loro difficile reperibilità. D'altra parte, i dispositivi MIDI hanno alcune limitazioni intrinseche, come specificato da (Loy, 1985), mentre le unità DSP, potendo essere programmate, permettono più flessibilità di utilizzazione e il loro costo sta diventando sempre più accessibile. In più, le crescenti capacità computazionali degli odierni Personal Computer general purpose permettono ormai l'utilizzo di schede DSP per applicazioni real time.

E' per questi motivi che noi abbiamo pensato che fosse arrivato il momento di integrare in una sola applicazione a finalità musicali le peculiarità del mondo MIDI e quelle del mondo DSP e che permettesse il controllo delle schede per il Signal Processing via codifica MIDI, aprendo quindi una significativa finestra sul controllo del suono in tempo reale, reso finalmente accessibile da un ambiente standardizzato e largamente diffuso.

In questo articolo mostreremo i risultati della nostra ricerca, condotta nell'ambito del progetto di ricerca Stazione di Lavoro Musicale Intelligente (SLMI) (Camurri et al., 1989; Stiglitz Ed., 1991), realizzata nell'ambito del Progetto Finalizzato "Sistemi informatici e calcolo parallelo" del C.N.R. e tesa ad evidenziare la possibilità di integrazione tra software commerciali ed alcuni prototipi realizzati presso il L.I.M. dell'Università di Milano e presso il D.I.S.T. dell'Università di Genova. La maggior parte dei prototipi realizzati è stata sviluppata per lavorare su macchine Macintosh. Tutte le applicazioni sono in grado di leggere e/o scrivere Standard MIDI File 1.0 nei differenti formati definiti. I SoundFile sono compatibili con il formato di Sound Designer II della Digidesign.

E' infine importante mettere in luce che la SLMI è un sistema aperto, al quale gli utenti possono aggiungere e togliere le applicazioni con cui sono abituati a lavorare allo scopo di trovare utili sinergie con i programmi prototipali della workstation, potendo comunicare tra le varie applicazioni tramite gli standard sopracitati.

2. Architettura generale dell'ambiente software

In questo articolo parleremo di tre applicazioni, SQAN (MIDI/DSP Sequencer), Waver Workbench (Sintesi mediante Two Variable Function, Waveshaping, Wavelet) e LASy Workbench (Sintesi mediante Automi Cellulari). Queste tre applicazioni sono in grado di scambiarsi informazione in modo intenso e atipico, costituendo un vero e proprio ambiente integrato per la costruzione, l'elaborazione e l'uso di suoni sintetizzati mediante DSP con la possibilità di esecuzione integrata a normali dispositivi MIDI.

Quando si parla di un insieme di tool software in grado di scambiarsi informazioni, è necessario innanzitutto chiarire di che tipo di informazione si tratta. Il nostro sistema si basa su tre standard diffusi: Standard MIDI Files 1.0 e MIDI Management Tools (per lo scambio di dati MIDI), Digidesign SoundFile (per lo scambio di informazione audio). Abbiamo inoltre definito un nuovo formato per lo scambio di parametri per la sintesi del suono tra le tre applicazioni che abbiamo chiamato Patch File.

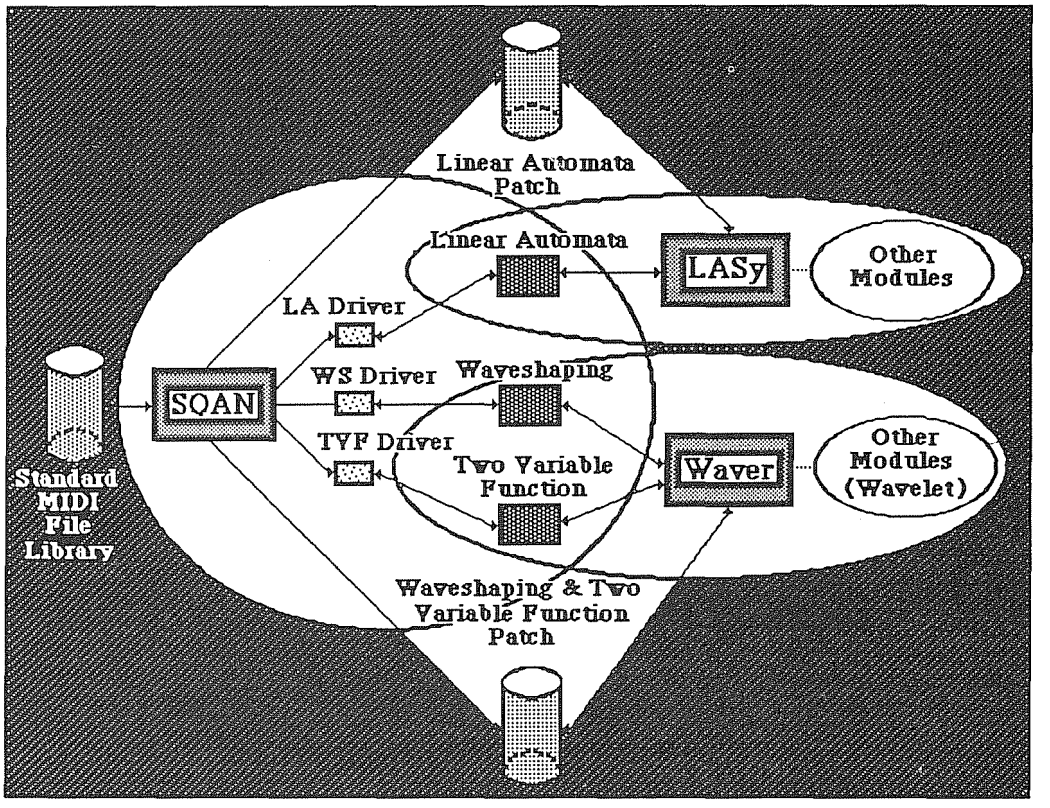


Figura 1

Tipo di linguaggi usati per la costruzione del sistema software:

- Linguaggi di alto livello per Macintosh (Pascal, C)
- Linguaggio di basso livello per Macintosh e DSP (Assembly 68000 e 56000)
- Linguaggio per DSP (Assembly 56000)

Come si può notare dalla Figura 1, SQAN, pur essendo un'applicazione distinta da LASy Workbench e da Waver Workbench, possiede alcune parti di codice che sono state integrate nella sua struttura come apparirà più chiaro in seguito.

Il sequencer può leggere e scrivere Standard MIDI Files 1.0 e Patch Files per lo scambio di informazioni con il mondo MIDI da una parte (rappresentato dalle ormai numerose applicazioni commerciali) e il mondo di ricerca e sviluppo rimbrico su DSP dall'altra (rappresentato nel nostro caso da Waver Workbench e LASy Workbench). Queste ultime due applicazioni possono inoltre leggere e scrivere Patch Files per creare ed elaborare suoni che potranno essere usati da SQAN. Così, nelle due applicazioni orientate allo sviluppo timbrico, potremo trovare in più rispetto al sequencer alcune features

particolari e strumenti di generazione delle tabelle utilizzate dalle sintesi e che non sono propriamente necessari in una applicazione orientata alla performance quale è SQAN. Poiché la configurazione del DSP dipende fortemente dal tipo di sintesi usata, diciamo per completezza che il Patch File ha un codice di identificazione che specifica il tipo si sintesi da cui è stato creato e da cui può quindi essere usato. Questo implica che mentre SQAN è in grado di leggere tutti i tipi di Patch Files, non c'é finora nessun buon motivo per cui ad esempio LASy Workbench, che gestisce la sintesi mediante Automi cellulari, sia in grado di leggere Patch Files creati per la gestione della sintesi mediante Waveshaping.

Infine possiamo dire che il sequencer è compatibile con il MIDI Managment Tools sotto MultiFinder e quindi, tra l'altro, consente di eseguire il brano musicale in uso anche se il MIDI/DSP Sequencer non è l'applicazione corrente realizzando una sorta di multitasking su Macintosh.

3. Il Sequencer MIDI/DSP (SQAN)

Il modulo Sequencer MIDI/DSP (SQAN) è un'applicazione, sviluppata in ambiente MPW su computer Macintosh, di tipo *sequencer* (registratore e processore di sequenze di dati musicali in formato MIDI) per il controllo simultaneo e sincrono di unità MIDI e di schede DSP e per il mixing a livello MIDI di sequenze musicali.

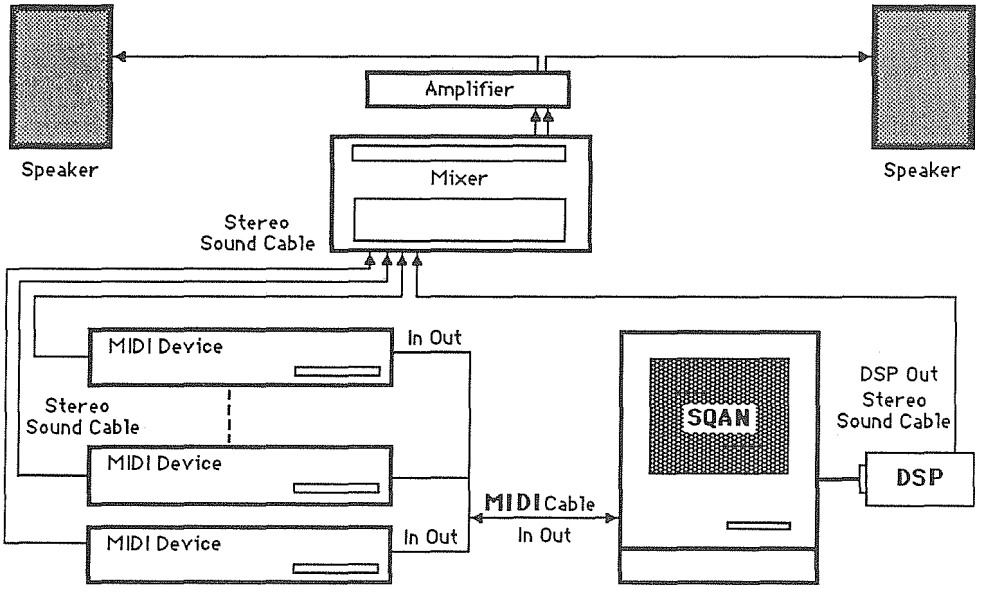


Figura 2

L'applicazione è composta da due sezioni: un *sequencer MIDI* che unisce alle consuete funzioni di registrazione, riproduzione ed elaborazione di sequenze MIDI anche la possibilità di controllare, tramite un apposito driver, da 1 a 4 schede Digidesign Sound Accelerator (basate sul DSP Motorola 56001)

collocate internamente al Macintosh, vedendole come normali unità di sintesi esterne controllabili via MIDI; un *mixer* per il bilanciamento dei livelli di intensità di differenti e contemporanee sequenze di dati musicali (numero massimo: 32) che opera direttamente sulle sequenze di dati MIDI, agendo sul parametro MIDI standard *key-velocity*, ed eliminando quindi la necessità di utilizzare per l'attività di mixing un apposito dispositivo hardware esterno di tipo convenzionale. In Figura 2 è mostrata l'architettura hardware del sistema.

Il Sequencer MIDI/DSP legge e scrive Standard MIDI Files 1.0 ed è compatibile con il MIDI Managment Tools della Apple Computer, potendo così scambiare dati MIDI con altre applicazioni aperte nel sistema tramite PatchBay, il pannello di controllo del MIDI Managment Tools.

In Figura 3 è infine riportato il pannello di controllo di una traccia dove si possono selezionare le unità di sintesi desiderate (MIDI e/o DSP).

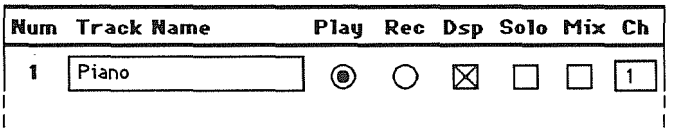


Figura 3

Ma tra tutte le caratteristiche di SQAN, quella di maggiore interesse attualmente è senz'altro la sua capacità di controllare via MIDI una o più schede DSP, vedendo queste ultime come normali sintetizzatori di suoni che però sono collocati internamente al calcolatore e sono programmabili.

3.1 Integrazione MIDI/DSP

Nel suo normale processo di lettura temporizzata dei dati MIDI che formano il brano musicale corrente, SQAN controlla se il messaggio MIDI appartiene o meno ad una traccia assegnata al DSP e in caso affermativo esegue una chiamata ad un driver per tradurre il significato del messaggio MIDI in termini in fraquenza ed intensità interpretabili dalle routines di controllo del DSP. Cosicché la prima cosa da fare parlando di integrazione MIDI/DSP è di definire le relazioni semantiche tra i comandi MIDI e le azioni del DSP. Ovviamente non tutti i messaggi MIDI sono da considerarsi interessanti per un primo approccio verso l'integrazione MIDI/DSP. Noi abbiamo per ora concentrato la nostra attenzione sui comandi MIDI di *Note On*, *Note Off* e *Program Change*.

E' abbastanza chiaro ed intuitivo che si possono definire le seguenti:

Note On -----> *Operazioni di inizio suono*

Note Off -----> *Operazioni di fine suono*

Una volta che SQAN ha passato un messaggio di *Note On* al driver DSP, è il driver stesso che, in dipendenza del tipo di sintesi in uso sulla scheda, deve mandare al DSP uno o più messaggi per generare un suono di una specifica frequenza (Key number) e dinamica (Key velocity). Il modo di calcolare la frequenza del suono dipende strettamente da tipo di sintesi che si sta usando ed è quindi opportuno disporre di un driver per ogni sintesi. Notiamo inoltre che la variazione del parametro di dinamica (Key

Velocity) solitamente causa un cambiamento nel contenuto armonico del suono prodotto; l'algoritmo di sintesi è ovviamente libero di trattare l'informazione di dinamica nel modo più opportuno.

Il messaggio di *Note Off* provoca l'interruzione di un suono di una specifica frequenza che è attivo sul DSP ponendo il volume della scheda a zero. L'implementatore della sintesi è anche qui libero di gestire eventuali altri parametri che possono essere in dipendenza di questo evento come velocità di Decay, Release etc.

Controllando il flusso dei Note On e dei Note Off, l'algoritmo di sintesi è in grado di conoscere attraverso il driver quale livello di polifonia è richiesto dalle tracce assegnate ad uno specifico DSP. In questo modo la sintesi potrà comportarsi correttamente ripetto al suo livello di polifonia.

Ora è importante definire le altre azioni della scheda afferenti al controllo dinamico dello spettro sonoro, inviluppo e variazione del suono da mandare in esecuzione. Queste azioni definiscono quello che comunemente indichiamo con il nome di "timbro".

Con il termine *Patch* noi facciamo riferimento all'insieme di dati e/o azioni che possono determinare cambiamenti della forma d'onda. In linea di principio un Patch può essere un insieme di parametri di sintesi, un identificatore di un algoritmo, una tabella o un'altra struttura dati da caricare sul DSP, un file di campioni. Così, quando vogliamo cambiare il "timbro" del suono corrente, dobbiamo mandare un messaggio, un messaggio MIDI, alla scheda. Poiché nel mondo MIDI esiste già un comando per cambiare timbro sui normali sintetizzatori e poiché noi vediamo il DSP come un sintetizzatore, abbiamo scelto il messaggio MIDI di *Program Change* per controllare l'aspetto timbrico del suono generato dalla scheda.

Possiamo così scrivere che:

Program Change -----> *Patch Change Operations*

Questa scelta implica due importanti situazioni: abbiamo la possibilità di indirizzare una tabella di 128 (7 bit) Patch (inclusi nel Patch file) senza dover cambiare la tabella stessa e, usando il messaggio MIDI di Program Change, siamo sicuri che tutti gli utenti di sistemi MIDI conoscono il significato del messaggio MIDI utilizzato per il cambiamento del "timbro".

Quindi, inserendo nella sequenza MIDI una opportuna successione di Program Change Messages tramite un apposito dialogo, si può gestire il comportamento del DSP dal punto di vista timbrico. E' importante sottolineare che al momento della ricezione del messaggio di Program Change, l'algoritmo di sintesi può, immediatamente e con qualsivoglia criterio, cambiare il suono sulla scheda anche se ci si trova tra un Note On e un Note Off; sugli usuali strumenti MIDI questa operazione è raramente implementata e in ogni caso di difficile impostazione. E' inoltre possibile creare e modificare librerie di "suoni", ognuna costituita da 128 possibili elementi, per configurare adeguatamente la scheda in dipendenza della sintesi che si intende usare. Le sintesi che sono state finora implementate sono: Two Variable Function Synthesis, WaveShaping Synthesis, Linear Automata Synthesis, come vedremo più in particolare nel § 5.

4. Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card

Il modulo Ambiente sperimentale per microprogrammi DSP per Sound Accelerator Card consente di costruire e modificare modelli timbrici basati su alcune tecniche specifiche di sintesi del suono (Wavelet, Waveshaping, TVF, Automi cellulari).

In particolare, è attualmente costituito da quattro sottoambienti, uno per ciascuna tecnica di sintesi implementata: uno per l'editing di modelli timbrici basati sulla tecnica di sintesi per Wavelet; un altro per l'editing di modelli timbrici basati sulla tecnica di sintesi per Waveshaping; un altro ancora per l'editing di modelli timbrici basati sulla tecnica di sintesi per funzioni a due variabili; infine l'ultimo, *LASy* (Linear Automata Synthesis) per l'editing di modelli timbrici basati su una tecnica di sintesi che utilizza automi cellulari come modelli per l'evoluzione dei suoni. I primi tre sottoambienti sono accessibili da una singola applicazione, Waver Workbench, mentre la tecnica LASy è resa disponibile dall'applicazione LASy Workbench. I modelli timbrici generati per mezzo delle tecniche di sintesi (a parte la tecnica Wavelet, come vedremo nel prossimo paragrafo) possono essere archiviati sotto forma di *Patch File*, un formato contenente informazione sul tipo di configurazione sonora predisposta per la scheda e che può essere letto dal Sequencer MIDI/DSP.

4.1 Waver Workbench

Waver Workbench è una applicazione nata per sperimentare tre tipi di sintesi:
- Two Variable Function Synthesis (TVF Synth). E' completamente descritta in (Mitsuhashi,1982) e sviluppata in (Borgonovo & Haus, 1986). E' una tecnica di sintesi non usata commercialmente, ma oggetto di ricerca da molti anni presso il L.I.M. Consiste essenzialmente nel campionamento di una funzione bidimensionale lungo un'orbita definita da un insieme di parametri.
- Waveshaping Synthesis (WSH Synth). E' completamente descritta in (Reinhard,1981) ed è ben conosciuta e usata commercialmente. Consiste nello "shaping" di un'onda sinusoidale, che viene poi utilizzata come argomento di un polinomio assegnato.
- Wavelet Transform Synthesis (WLT Synth). E' nata principalmente come tecnica di analisi simile alle Fourier Transform (Kronland-Martinet, 1988). La WLT può anche essere usata per sintetizzare suoni poiché, dopo il calcolo della trasformata, esistono molte regole di ricostruzione che permettono di resintetizzare il suono dopo che su di esso sono state eseguite interessanti manipolazioni.

Waver Workbench è un'applicazione utile sia per realizzare ricerca sul comportamento delle sintesi implementate, sia per realizzare, oltre ai Patch File, altre tabelle che è possibile usare anche dall'ambiente Sequencer MIDI/DSP per la produzione di suoni. Per quanto riguarda in particolare le sintesi TVF e WSH è per entrambe possibile, tramite un apposito dialogo, cambiare in tempo reale i parametri delle due sintesi ascoltando direttamente il risultato delle modifiche. Lanciando Waver possiamo accedere direttamente a tutte e tre le sintesi potendo inoltre visualizzare i suoni sintetizzati per mezzo della scrittura di SoundFile Digidesign.

Lo sviluppo di Waver è avvenuto in parallelo con lo sviluppo di SQAN e l'integrazione nel Sequencer MIDI/DSP di due delle sintesi di Waver (TVF e WSH) ha rappresentato il primo esperimento sul quale è stata definita l'intera strategia di integrazione. L'esclusione da SQAN della sintesi tramite Wavelet è avvenuta poiché erano necessari tempi di calcolo troppo elevati per essere eseguiti in tempo reale su un hardware come quello usato per la realizzazione del prototipo e in questa sede ha contribuito a chiarire i limiti di calcolo del prototipo stesso. E' apparso chiaro che l'architettura software di SQAN, con l'hardware disponibile in questa prima implementazione, era più adatta all'integrazione di metodi di sintesi algoritmica che all'implementazione di metodi di trasformazione di campioni che implicavano, nel nostro caso specifico, il trasferimento di una grande mole di dati.

4.2 LASy Workbench

La Linear Automata Synthesis è una nuova tecnica di sintesi che è completamente descritta in (Chareyron,1990). L'algoritmo LASy "vede" una tabella contenente un'onda campionata come un

automa cellulare unidimensionale. Lo stato di ogni cella dell'automa è periodicamente calcolato utilizzando la sua regola di transizione. I valori corrispondenti sono inviati al DAC, permettendo la generazione di una vasta varietà di suoni in grado di automodificarsi.

Lasy Workbench è un sistema software che implementa l'algoritmo LASy. E' un'applicazione orientata sia alla ricerca che all'uso pratico. LASy Workbench permette ai musicisti di sperimentare la sintesi sonora e di creare librerie di strumenti software che usano l'algoritmo LASy e che possono essere usate dal sequencer SQAN. L'applicazione può anche essere usata come un modulo MIDI dedicato alla produzione di suoni permettendo su questi ultimi un controllo real-time.

Al momento del lancio dell'applicazione, vengono caricate automaticamente alcune tabelle di default per il controllo delle forme d'onda e delle regole di transizione, per consentire all'utente un uso immediato della sintesi LASy. La generazione del suono è controllata dalla tastiera del computer (come fosse una tastiera musicale nel senso comune del termine) o da un dispositivo MIDI che agisce esternamente che viene connesso al calcolatore tramite una interfaccia MIDI. Lo schermo mostra la maggior parte dei parametri della sintesi che possono essere modificati usando il mouse o la tastiera del calcolatore. I files di forme d'onda, regole di transizione e "strumenti" possono essere letti e scritti su hard disc.

Diversamente dallo sviluppo di Waver Workbench, sviluppato parallelamente a SQAN e già con una certa attenzione rivolta al problema della integrazione dei microprogrammi nel sequencer, LASy Workbench è nato come una applicazione stand-alone e l'integrazione della tecnica LASy è cominciata dopo il completamento di SQAN. Inoltre TVF and WSH sono due tecniche di sintesi che presentano molte caratteristiche in comune mentre LASy ha, come abbiamo visto, una natura completamente differente. Questo fatto ha, come vedremo in seguito, contribuito a collaudare le specifiche di integrazione di altre sintesi all'interno del Sequencer MIDI/DSP.

5. Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card

Il modulo Microprogrammi per Sequencer MIDI/DSP e Sound Accelerator Card consiste in una libreria aperta di microprogrammi che implementano attualmente le seguenti teniche di sintesi: TVF, Waveshaping e Automi Cellulari.

5.1 Il modulo TVF del sequencer SQAN

Esaminiamo qui brevemente come la sintesi TVF (Two Variable Function) è stata integrata nel Sequencer MIDI/DSP.

Un microprogramma scritto in assembly 56000 realizza la sintesi in tempo reale campionando una tabella che contiene una funzione a due variabili su un'orbita definita da 10 parametri la quale può essere modificata per variare il timbro.

A causa della complessità dell'algoritmo, la sintesi è stata implementata soltanto in una versione monofonica; infatti abbiamo riscontrato problemi di tempo di calcolo dovuti al fatto di dover calcolare un singolo campione in meno di 1/44100 sec, che è il tempo che intercorre tra due interrupt successivi generati dalla porta SSI connessa al DAC della scheda Sound Accelerator. Per quanto riguarda la gestione dell'inviluppo, abbiamo usato un modello semplice, basato su tre parametri: volume, velocità di attacco e velocità di rilascio. Ciò ha permesso di semplificare la parte preposta al controllo dell'inviluppo orientando maggiormente il microprogramma verso il controllo del timbro.

In Figura 4 vediamo il dialogo principale di Waver per la sintesti TVF.

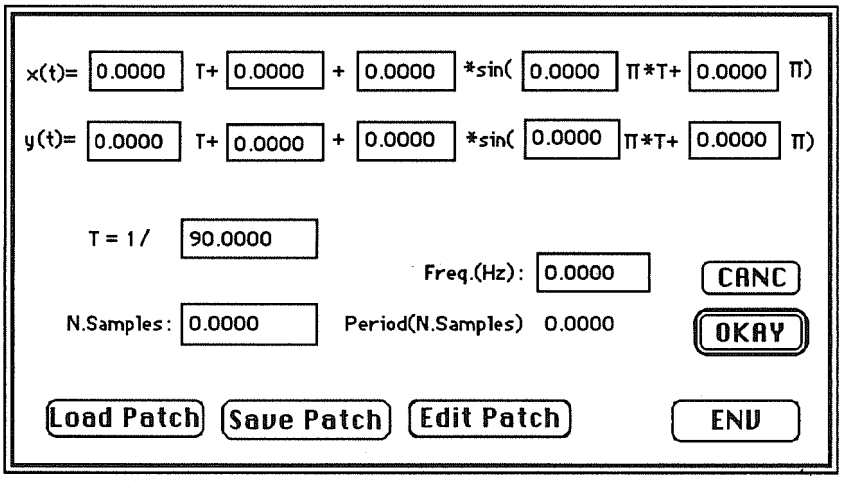


Figura 4

I tre pulsanti presenti nella parte inferiore del dialogo sono comuni a tutti gli algoritmi di sintesi inclusi in SQAN e hanno la stessa azione. La pressione sul pulsante "Save Patch" (salvataggio di un Patch) fa apparire il relativo dialogo di SQAN. Quando l'utente segnala l'accettazione dell'operazione di salvataggio del patch (dopo aver eventualmente modificato il nome proposto dal dialogo), il patch attivo (che corrisponde alla lista dei parametri visibili nel dialogo pricipale della sintesi LASy) viene salvato in memoria, e può ulteriormente essere ricaricato o modificato. L'insieme dei patch in memoria e le relative tabelle possono essere memorizzati in una delle memorie di massa in linea con la scelta della voce "Save Patch" del menu "File" di SQAN. La pressione sul pulsante "Load Patch" (caricamento di un patch) fa apparire il relativo dialogo di SQAN. Quando l'utente segnala l'accettazione dell'operazione di caricamento il patch attivo viene sostituito dal patch scelto. Il dialogo principale della sintesi LASy viene immediatamente aggiornato in modo che i valori visualizzati corrispondano a quelli dei parametri del patch appena caricato. La pressione sul pulsante "Edit Patch" (caricamento di un patch) fa apparire il relativo dialogo di SQAN. L'utente ha la possibilità di modificare il nome di un patch in memoria, o di sopprimerlo del tutto.

La struttura del file di Patch per la sintesi TVF vede:
- la *parte statica* costituita dalla tabella contenente la funzione di due variabili che viene spedita solo all'atto dell'apertura della sintesi;
- la *parte dinamica* costituita da un record contenente i 10 parametri di sintesi, i 3 parametri dell'inviluppo, il parametro T.

Questo record realizza il Patch e contiene i dati che vengono mobilitati ad ogni Program Change, che si realizza nella spedizione dei soli parametri cambiati rispetto ad un precedente Patch .

Nella parte alta del dialogo compaiono le due espressioni delle coordinate dell'orbita in cui ogni parametro è stato implementato come 'parametro freccia':

$$x(t) = f_X T + p_X + I_X \sin(\varphi_X \pi T + \psi_X \pi)$$
$$y(t) = f_Y T + p_Y + I_Y \sin(\varphi_Y \pi T + \psi_Y \pi)$$

In particolare:

- f_X, f_Y, φ_X, φ_Y: sono parametri freccia di *passo 1* moltiplicati per il parametro T
- p_X, p_Y, ψ_X, ψ_Y: sono parametri freccia di *passo T*
- T è uguale a 1/TableSize con TableSize parametro freccia di *passo 1*.

Tramite quest'ultimo si può agire sull'intonazione effettuando una sorta di accordatura dello strumento.
Ogni volta che viene cambiato un parametro si vedrà comparire un valore in corrispondenza dei due parametri Freq(Hz) e Period(N.Samples); questi sono rispettivamente il valore suggerito per la frequenza e per il periodo espresso in numero di campioni prima del ripetersi della forma d'onda. Nel caso in cui il calcolo fosse inesatto, è possibile indicare il valore corretto agendo sul *parametro editabile* Freq(Hz), indicandone il vero valore. Questa operazione ha importanza quando si opera nell'ambiente integrato MIDI/DSP dove salvare un Patch comporta anche la memorizzazione del parametro T su cui si baserà il calcolo di trasposizione della frequenza in fase di esecuzione delle note.

N.Samples è un *parametro editabile* intero avente significato solo quando il dialogo è richiamato con abilitata un opzione per la produzione di un SoundDesigner File per esaminare il risultato della sintesi. Il bottone "Env" permette di accedere al *dialogo di cambio dell'inviluppo* in cui si possono controllare i parametri di *volume*, *velocità di attacco* e *velocità di rilascio*. Il bottone "Canc" permette di uscire dal dialogo di cambio dei parametri senza però attualizzare gli stessi; mentre il bottone "Ok" esegue l'uscita dal dialogo più attualizzando le variazioni apportate.

5.2 Il modulo WSH del sequencer SQAN

Nella nostra implementazione, la sintesi WSH è stata vista come un particolare caso monodimensionale della sintesi TVF, dove x=i permette la scelta del polinimio p_i memorizzato nella riga i_{-esima} della tabella dei polinomi mentre

$$y(t) = I \sin(2\pi ft+p)$$

è l'onda di input che deve essere distorta. E' importante notare che il cambiamento da un polinomio ad un altro può generare discontinuità e così eseguiamo questa operazione solo al verificarsi di un Program Change e dopo aver eseguito i controlli del caso.

Il programma in microcodice che realizza la sintesi WSH è una restrizione di quello della sintesi TVF: la differenza risiede nella tabella, la quale è generata con un criterio differente. In particolare la sintesi TVF, usata con una tabella WSH, può generare waveshaping avendo l'attenzione di lasciare inalterati tutti i parametri eccetto i seguenti:

$f_X = i$ (indice intero del polinomio)
$I_Y \in [-1,1]$ invece di $I_Y \in \Re$
$\psi_Y \in \Re$

In conseguenza di ciò, sia la sintesi WSH che la sintesi TVF possono essere classificate come tecniche di Not Linear Distortion (NLD); nel caso della sintesi TVF la previsione dei valori di output risulta sicuramente più complessa poiché sia la funzione di input che la Transfer Function sono entrambe in due variabili e in piu, nel caso TVF, è presente il concetto di "wrap around" che non si presenta nel caso WSH, essendo $I_y \in [-1,1]$. Riguardo al problema dell'inviluppo, abbiamo usato gli stessi accorgimenti implementati per la sintesi TVF.

Il dialogo di controllo della sintesi WSH non viene riportato poiché è molto simile come tipo di interazione e come parametri a quello della sintesi TVF.

5.3 Il modulo LASy del sequencer SQAN

Sono necessari al minimo due moduli software per integrare una nuova tecnica di sintesi nel sequencer SQAN: un microprogramma in assembly 56000 caricato sulla scheda della Digidesign preposto a calcolare i campioni che è necessario fornire al DAC della scheda stessa e un driver che traduce i messagi MIDI passati da SQAN in appropriati comandi da mandare alla Sound Accelerator Card via Nubus. Un terzo modulo, il modulo di interfaccia, diviene necessario quando si volesse avere una comunicazione diretta con l'utilizzatore per consentire editing dei suoni dall'interno dell'applicazione sequencer. Il modulo LASy, come del resto i moduli TVF e WSH, permette questa interazione.

Il microprogramma per l'algoritmo LASy consente di disporre di una sintesi sonora polifonica (sono disponibili quattro voci nella attuale implementazione). Siamo stati piacevolmente sorpresi dal fatto che il microprogramma polifonico implementato per LASy Workbench sia stato utilizzato per SQAN senza dovergli apportare alcuna modifica!

Ogni oscillatore software è associato a due tabelle collocate nella memoria della scheda che rappresentano i due stati più recenti dell'automa. Il loop principale del microprogramma aggiorna queste due tabelle seguendo la regola di transizione corrente, la quale può essere diversa per ogni oscillatore. Questo loop è interrotto con una frequenza pari a quella di campionamento per fornire il campione al DAC. La routine di interrupt SSI preleva il corretto campione dalle tabelle in funzione della frequenza dell'oscillatore, usando una interpolazione lineare per compensare la lunghezza limitata delle tabelle stesse. I campioni di ogni oscillatore vengono quindi scalati in funzione del volume corrente, sommati ed infine mandati al DAC. Il loop principale controlla inoltre i flag che indicano la richiesta di esecuzione di una nuova nota (per caricare una nuova forma d'onda e settare il volume corrente), di un messaggio di Note Off (passando alla regola di transizione per il rilascio) o la richiesta di fine suono (azzerando il volume degli oscillatori).

La parte più complessa del lavoro di integrazione è stata rappresentata dalla gestione delle tabelle (Patch Table) di suoni, poiché ogni operazione richiede numerosi passaggi verso le rountines di controllo di SQAN. Ciò è dovuto in gran parte al fatto che é stato costruito un vero e proprio protocollo di comunicazione tra la sintesi e il sequencer per consentire ad altri sviluppatori di aggiungere altre sintesi. Essendo questo un protocollo standard, si può adattare in modo più o meno naturale alla sintesi che si vuole implementare. Questo è il prezzo da pagare per presentare una interfaccia consistente ad un utente interessato ad usare diverse sintesi per la produzione del suono e nel suo complesso tutto ciò sembra più che adeguato. Trasferimento delle Patch a parte, tutte le altre funzionalità si sono incastonate bene nel telaio predisposto per la loro realizzazione, essendo disponibile anche una buona documentazione a riguardo dell'intero protocollo di comunicazione tra il sequencer e la sintesi. Vediamo ora in Figura 5 i parametri di controllo della sintesi.

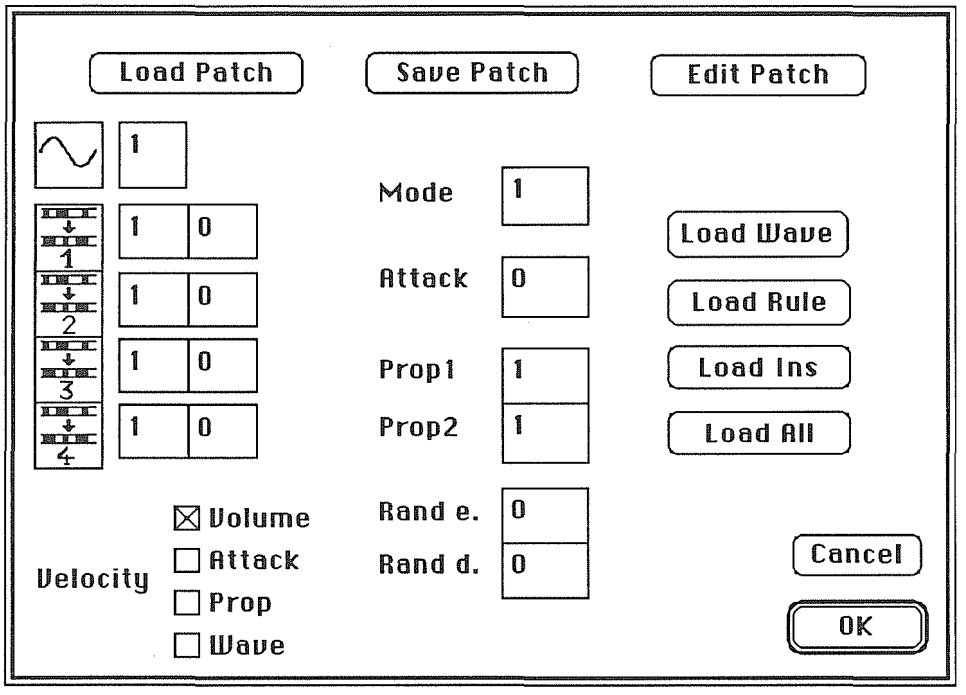


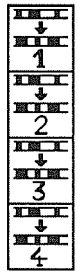
Figura 5

Questo dialogo permette di precisare tutti i parametri che determinano un patch e di lanciare le operazioni sui file di dati. In basso a destra troviamo i pulsanti "OK" e "Cancel" usati per chiudere il dialogo:
- il pulsante "Cancel" chiude il dialogo senza modificare il patch corrente. I valori dei parametri modificati nel dialogo vengono ignorati;
- il pulsante "OK" chiude il dialogo e modifica il patch corrente a partire dai valori dei parametri al momento della chiusura. Se il valore dato ad un parametro non è valido, il dialogo viene subito riaperto per permettere di correggere l'errore commesso.

Le opzioni "Load Wave", "Load Rule", "Load Ins", "Load All" sono raggruppate alla destra del dialogo principale di LASy, e servono a caricare un diverso tipo di file creato nell'ambiente sperimentale LASy Workbench.

Il campo "Numero della forma d'onda" (a destra dell'icona ) contiene il numero d'ordine (all'interno della tabella presente in memoria) della forma d'onda attiva. Questa forma d'onda viene caricata nella scheda DSP all'inizio di ogni nota per essere in seguito modificata secondo la/le regole di transizione attive.

I campi "Numero della regola di transizione" (immediatamente a destra delle icone



) contengono i numeri d'ordine (all'interno della tabella presente in memoria) delle regole di transizione attive. Un massimo di quattro regole di transizione possono essere attive in contemporanea per modificare la forma d'onda e produrre il timbro desiderato. Il ruolo di ciascuna di queste regole dipende dal modo di funzionamento scelto (campo "Mode") come spiegato più avanti. La modifica di una regola di transizione non pertinente al modo corrente non ha nessun effetto.

I campi "Numero dell'algoritmo" usato danno la regola di transizione corrispondente" (a destra dei quattro campi precedenti).

Questi quattro campi (che corrispondono alle quattro regole di transizione attive) contengono il numero dell'algoritmo usato per calcolare il valore di un nuovo campione a partire dalla generazione precedente della forma d'onda e dei valori della tabella della regola di transizione. Nella corrente implementazione della sintesi LASy all'interno di SQAN (versione 1.0) l'unico valore pertinente è 1, che corrisponde al calcolo del nuovo campione, basato sulla somma dei valori del vecchio campione e dei suoi due vicini.

Al centro del dialogo troviamo i campi dedicati agli altri parametri della sintesi.

Il campo "Mode".

Il valore del parametro "Mode" determina il modo nel quale le regole di transizione attive contribuiscono all'evoluzione della forma d'onda. Nella versione attuale (1.0) il parametro "Mode" può prendere 4 valori :
- Mode 1: la regola di transizione n.1 entra in funzione all'inizio della nota. Quando il messaggio di fine nota corrispondente viene ricevuto, la regola di transizione n.2 prende il suo posto;
- Mode 2: La regola di transizione n.1 rimane attiva per un numero di generazioni uguale al valore del campo "Attack" (eventualmente modificato secondo il valore di velocity della nota), dopodiche viene sostituita dalla regola di transizione n.2. Quando il messaggio di fine nota corrispondente viene ricevuto, la regola di transizione n.3 prende il suo posto;
- Mode 3: Le regole di transizione n.1 e n.2 entrano in azione con un peso proporzionale ai valori dei campi "Prop 1" e "Prop 2". Quando il messaggio di fine note corrispondente viene ricevuto, la regola di transizione n.3 prende il loro posto;
- Mode 4: La regola di transizione n.1 rimane attiva per un numero di generazioni uguale al valore del campo "Attack" (eventualmente modificato secondo il valore di velocity della nota), dopodiche viene sostituita dalle regole di transizione n.2 e n.3 con un peso proporzionale ai valori dei campi "Prop 1" e "Prop 2". Quando il messaggio di fine note corrispondente viene ricevuto, la regola di transizione n.4 prende il loro posto.

Il campo "Attack".

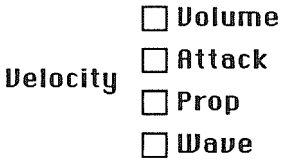
Il valore del campo "Attack" ha significato nei modi di sintesi 2 e 4. Il suo ruolo è descritto nella discussione del campo "Mode".

I campi "Prop1" e "Prop2".

Il valore dei campi "Prop1" e "Prop2" hanno significato nei modi di sintesi 3 e 4. Il loro ruolo è descritto nella discussione del campo "Mode".

Il valore dei campi *"Random extension"* e *"Random deepness"* determina il modo in cui le note che corrispondono allo stesso patch vengono suonate. Se il valore di questi due parametri è zero, tutte le note eseguite con lo stesso valore di "velocity" seguono esattamente la stessa evoluzione armonica. Impostando valori non nulli, piccole variazioni vengono portate alla forma d'onda di partenza per evitare la ripetizione meccanica del suono generato. Il valore del parametro "Random extension" determina il numero complessivo di queste variazioni, mentre il valore del parametro "Random deepness" regola la loro ampiezza. Bisogna notare che gli stessi valori dei parametri "Random extension" e "Random deepness" hanno effetti diversi quando si cambia la regola di transizione corrente.

I campi "Velocity"



Non si tratta di campi numerici, ma di "check box" che possono essere attivate o meno per indicare in che modo il valore di Velocity del messaggio MIDI ricevuto agisce sulla sintesi del suono:
- check box "Volume": il valore di Velocity modifica il volume del suono emesso;
- check box "Attack": il valore di Velocity modifica il valore effettivo del parametro Attack nei modi di sintesi 2 e 4;
- check box "Prop": il valore di Velocity modifica il valore effettivo dei parametri "Prop1" e "Prop2" nei modo di sintesi 3 e 4;
- check box "Wave": il valore di Velocity modifica i valori della tabella di forma d'onda corrente.

Ringraziamenti

Ringraziamo Pier Luigi Bettitoni per lo sviluppo della funzionalità di mixing MIDI e della libreria di funzioni di import/export di Standard MIDI File 1.0. Inoltre, un grazie speciale allo staff del L.I.M. - Laboratorio di Informatica Musicale, per il supporto scientifico e tecnico costantemente fornito durante tutta la nostra ricerca.

Riferimenti bibliografici

A. Borgonovo, G. Haus: "Sound Synthesis by Means of Two-Variable Functions: Experimental Criteria and Results", **Computer Music Journal**, Vol.10, N.3, MIT Press, Cambridge, USA, 1986, pp.57-71.

A. Camurri, G. Haus, A. Stiglitz, R. Zaccaria, **LRC C4 MUSIC: architettura, specifiche funzionali di alto livello e standard di documentazione della STAZIONE DI LAVORO MUSICALE INTELLIGENTE**, Tech. Report 7/09, Progetto Finalizzato "SISTEMI INFORMATICI E CALCOLO PARALLELO", Consiglio Nazionale delle Ricerche, novembre 1989.

J. Chareyron, "Digital Synthesis of Self-Modifying Waveforms by Means of Linear Automata", **Computer Music Journal**, Vol.14, N.4, 1990, pp.25-41.

R. Kronland-Martinet, "The Wavelet Transform for Analysis, Synthesis and Processing of Speech and Music Sounds", **Computer Music Journal**, Vol.12, N.4, 1988, pp.11-20.

G. Loy, "Musician makes a Standard: the MIDI Phenomenon", **Computer Music Journal**, Vol.9, N.4, 1985, pp.8-26.

Y. Mitsuhashi, "Audio Signal Synthesis by Function of Two Variables", **JAES**, Vol.30, No.10, 1982, pp.701,706.

P. Reinhard, "Distorsione non lineare della somma di cosinusoidi: analisi dello spettro tramite matrici", **Atti del IV Colloquio di Informatica Musicale**, Pisa, 1981, pp.160-183.

AA.VV. (A. Stiglitz editor), **Specifiche funzionali di alto livello dei moduli della STAZIONE DI LAVORO MUSICALE INTELLIGENTE - Unità Operativa dell'Università degli Studi di Milano**, Tech. Report 7/53, Progetto Finalizzato "SISTEMI INFORMATICI E CALCOLO PARALLELO", Consiglio Nazionale delle Ricerche, marzo 1991.

HARP: UN AMBIENTE AD ALTO LIVELLO PER L'AUSILIO ALLA COMPOSIZIONE

A.Camurri, C.Canepa, M.Frixione, C.Innocenti, C.Massucco, R.Zaccaria

DIST - Universita di Genova
Laboratorio di Informatica Musicale
Via Opera Pia 11/A - 16145 Genova
posta elettronica: music@dist.dist.unige.it

Abstract

This paper describes a methodological framework and a prototype high-level system called HARP (Hybrid Action Representation and Planning) [3, 4]. HARP is designed to store and process music and sounds, and to carry out plans for manipulating this material according to the composer's goals. Therefore, it is able to help in the development of new music pieces as well as in manipulating existent ones, on the basis of given material: to this end, the system provides also formal analysis capabilities on both music and sounds, for extracting information useful in subsequent synthesis processes. In other words, HARP is a system for the representation and manipulation of music knowledge as regards composition: this is a vast field of knowledge which can be suitably managed by a twofold formalism. The sound itself (samples, codes, algorithms), real-time performance, particular analysis processes, are managed by procedures, included in an object-oriented concurrent environment (HARP *analogical subsystem*). Higher level scores, composition rules, definitions in general, descriptions of pieces of music, are stored in a declarative symbolic environment (HARP *symbolic subsystem*), based on a multiple-inheritance semantic network formalism derived from KL-ONE [2], that we have extended to represent *time*, a fundamental aspect of music knowledge [5, 6]. A formal link between the two subsystems exists, so that, for example, asking the semantic network to generate a particular instance of a music object, automatically "fires" the appropriate procedures at the analogical level.
A software prototype has been implemented using Arity Prolog and Microsoft C on PC-IBM under the Microsoft Windows SDK environment.

1. Introduzione

Sistemi informatici per l'ausilio alla composizione sono generalmente basati su strumenti e linguaggi per la manipolazione a basso livello di suoni, partiture ed algoritmi [1]. Recenti proposte basate su linguaggi orientati agli oggetti [7] e su tecniche di intelligenza artificiale [5, 6] sono orientate alla definizione di formalismi più efficaci e ad alto livello.

Nell'articolo viene introdotto un formalismo basato su tecniche di Intelligenza Artificiale, denominato HARP (Hybrid Action Representation and Planning), per la memorizzazione e la manipolazione ad alto livello sia di musica (partiture, descrizioni a diversi livelli di astrazione del materiale musicale) che di suoni. Tale formalismo è inoltre in grado di gestire la pianificazione di azioni complesse per la manipolazione di suoni e musica in funzione di obiettivi specificati dal

compositore.
L'articolo descrive brevemente anche l'architettura generale di un prototipo software basato su tale formalismo. In definitiva, il sistema basato su HARP (da ora in avanti semplicemente HARP) è in grado di generare, o di aiutare il compositore a sviluppare nuovi brani musicali, così come di manipolarne esistenti. Il sistema fornisce inoltre strumenti formali per l'analisi musicale, per l'analisi di materiale sonoro e per l'estrazione di informazioni utili in successive fasi di sintesi.

2. Descrizione generale ed introduzione al formalismo

HARP è un formalismo per la rappresentazione e manipolazione di conoscenza musicale riguardante l'ambito compositivo: si tratta di un vasto campo di conoscenza, che viene gestito dal sistema mediante la integrazione di due formalismi complementari.
Il suono (in forma di campioni, codice, algoritmi), performance in tempo reale, procedimenti particolari di analisi, sono gestiti da procedure, in un ambiente *object-oriented concorrente* (HARP *analogical subsystem*, il sottosistema analogico di HARP).
Partiture ad alto livello, regole compositive, descrizioni di forme musicali, definizioni in generale, sono memorizzate in un ambiente *simbolico dichiarativo* (HARP *symbolic subsystem*, il sottosistema simbolico di HARP), basato su di un formalismo a reti semantiche ad eredità multipla derivato dal KL-ONE [2] ed a sua volta strutturato in due componenti: la componente terminologica e quella asserzionale, descritte nel seguito.
Un collegamento formale è definito tra i due sottosistemi in modo tale che, ad esempio, una richiesta alla rete semantica riguardante la generazione di un particolare oggetto musicale causi l'attivazione delle appropriate procedure a livello analogico.

Prima di proseguire con la descrizione dei sottosistemi in cui si compone HARP, vale la pena accennare ad alcuni concetti basilari su cui si fonda la parte simbolica del sistema.

Come per tutti i linguaggi della famiglia KL-ONE e come generalmente assunto nella intelligenza artificiale, il termine *object-oriented* comporta differenze sostanziali rispetto l'uso assunto nei tradizionali linguaggi (Smaltalk, C++, ecc.).
In particolare, nel nostro contesto, il meccanismo noto come *eredità* si basa sul concetto di *specializzazione* e non in quello di *aggregazione*. Ad esempio, non è corretto definire un oggetto musicale *Score1* come l'oggetto che eredita le proprietà dagli oggetti *NoteList* ed *Instrument* (cioè definito come l'*aggregazione* delle proprietà ereditate dai due differenti oggetti). Questo perchè *Score1* non è nè una lista di note nè uno strumento.
Ha invece senso definire, ad esempio, *canone doppio retrogrado* come l'oggetto che eredita le sue proprietà da *canone doppio* e da *canone regtrogrado*, cioè ne è la *specializzazione* di entrambi.
Si noti che il formalismo consente eredità multipla, come risulta dall'ultimo esempio.
Questa interpretazione del meccanismo di eredità consente una pulizia formale ed una leggibilità e gestione migliore di grosse basi di conoscenza (con elevato numero di oggetti correlati fra loro); è inoltre considerato più fedele ad un possibile modello della organizzazione della conoscenza nella mente umana [8].

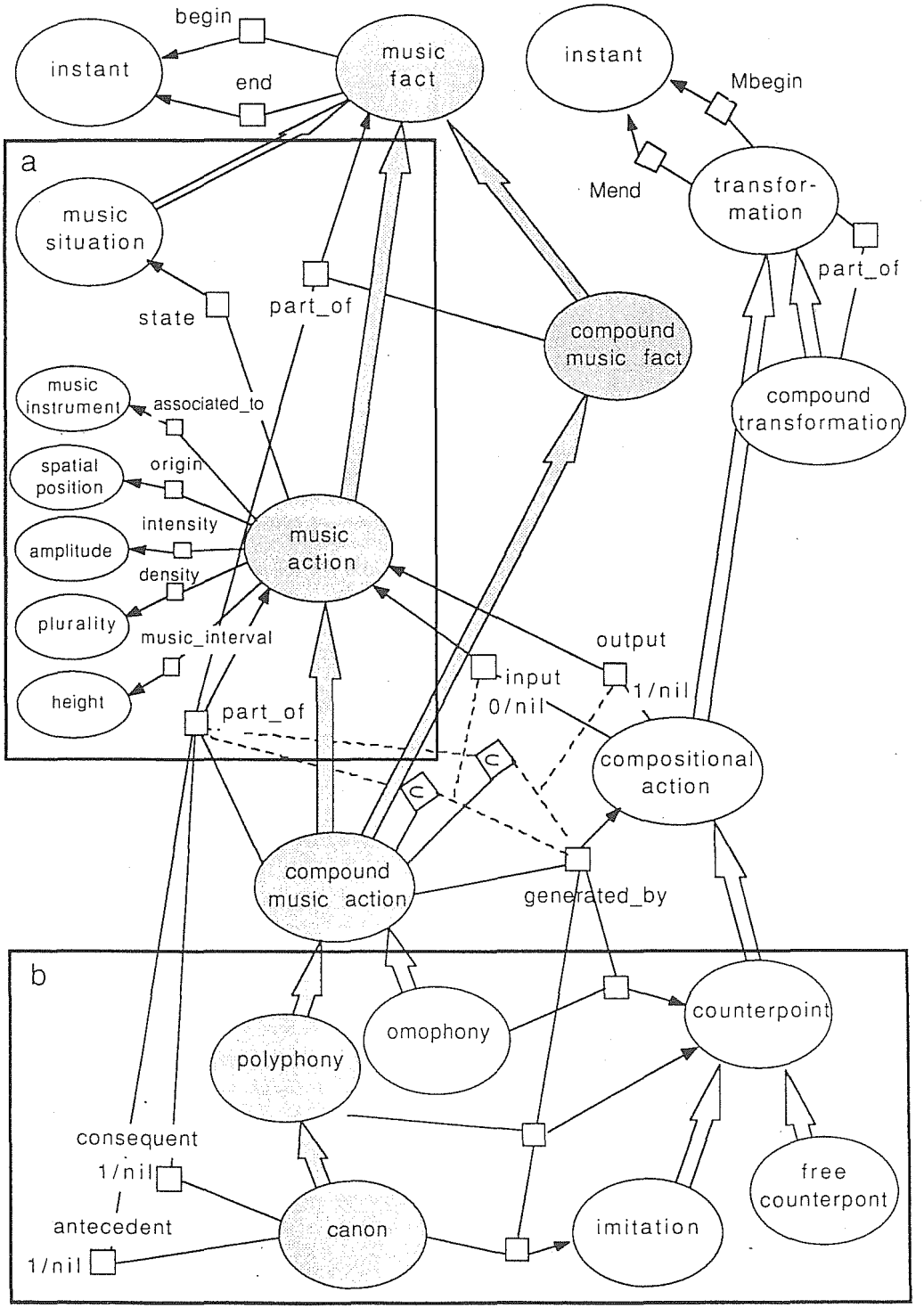


Figure 1: HARP symbolic subsystem: the terminologic component (excerpt)

3. HARP: Il sottosistema simbolico

Formalismi della famiglia delle reti semantiche sono già stati studiati ed utilizzati in ambito musicale: un esempio significativo è costituito dalle *DH-networks* [10], nate con l'intento di definire un modello di rappresentazione dell'ascolto, in grado di tener conto degli aspetti dinamici, strutturali, gerarchici tra *oggetti musicali.*

Il nostro approccio si basa su di un formalismo di tipo KL-ONE [2]. KL-ONE ha dato origine ed influenzato significativamente, negli ultimi anni, lo sviluppo di diversi sistemi e linguaggi per la rappresentazione della conoscenza e si può considerare come il formalismo a reti semantiche attualmente più diffuso [12]. In particolare, il sottosistema simbolico di HARP utilizza una estensione del KL-ONE verso il trattamento del *tempo*, aspetto fondamentale in musica [5, 6].

In figura 1 è mostrata una parte della base di conoscenza simbolica di HARP: ellissi corrispondono a concetti nella tassonomia, doppie frecce indicano *IS-A* link ("*e` un...*", eredità intesa come specializzazione); archi singoli corrispondono a *ruoli* (o *slots*, relazioni tra coppie di concetti). Ad esempio, una *music action IS-A music fact*, ed uno dei suoi ruoli è *intensity*, il cui valore è *amplitude* (cioè, la *intensity* di una azione musicale è una *amplitude*: figura 1, box "a").

La parte di base di conoscenza descritta in figura 1 è relativa alla componente *terminologica* del *symbolic subsystem*, cioè la parte relativa alle definizioni delle entità manipolate dal sistema, analogo delle *classi* in un linguaggio *object-oriented.* L'utente definisce tali entità allo scopo di crearne successivamente *istanze*, corrispondenti ad oggetti musicali reali. Le istanze sono memorizzate in una parte della base di conoscenza simbolica separata da quella terminologica, denominata *componente asserzionale.*

Due entità musicali basilari sono definite nella componente terminologica del sottosistema simbolico: la *music action* (azione musicale) e la *compositional action* (azione compositiva) (vedi figura 1).
La prima rappresenta materiale musicale a differenti livelli di astrazione: da un livello sonologico (descrizioni a basso livello di suoni), a descrizioni ad alto livello (ad esempio, nella musica classica occidentale, la definizione di *forme musicali* come la fuga o la sonata).
compositional action si riferisce ad una classe di meta-azioni, cioè ai possibili "manipolatori" che operano su azioni musicali (sia a livello di classi che di istanze) per produrre nuove azioni musicali, o per eseguire processi di analisi utili in successive manipolazioni. La classe delle *compositional action* consente al compositore di manipolare azioni musicali a seconda dei suoi attuali obiettivi: ad esempio, può operare su di una azione musicale a livello sonologico per definire, aggiustare particolari parametri del suono; durante la pianificazione della struttura globale di una parte, invece, la stessa azione musicale può essere vista come un oggetto più astratto, in relazione con eventuali altre azioni musicali. Il compositore può introdurre nel sistema sia sottoclassi che istanze di azioni musicali, rispettivamente nella componente terminologica od in quella asserzionale della base di conoscenza. Le prime sono oggetti riusabili, scheletri di partiture o definizioni di suoni; le *istanze* sono invece oggetti completamente definiti, ascoltabili tramite i canali sonori di uscita di HARP.

HARP è un ambiente che non si riferisce ad un particolare stile o contesto musicale. Ogni contesto musicale può richiedere la creazione di opportune definizioni nella parte simbolica, e delle relative procedure analogiche. Ad esempio, in un contesto tonale, è possibile definire la sottoclasse

canon, con la sua struttura: le sue azioni musicali componenti *antecedent* e *consequent* (box "b" di figura 1). Successivamente, il compositore può introdurre nella componente asserzionale, o chiedere al sistema di generare (o aiutarlo a sviluppare) un particolare canone, caratterizzato da un dato antecedente ed un "opportuno" conseguente.

Le caratteristiche basilari delle *music action* includono: (i) una connotazione temporale (i ruoli *begin* ed *end*), che consentono al compositore di correlare azioni musicali tra loro nel tempo; (ii) un insieme di relazioni, come l'evoluzione dinamica, il contenuto timbrico, melodico, metrico, ritmico. Supponiamo che un compositore voglia specificare in una *compositional action* che una *music action* (un frammento di un brano) *N* venga generata in correlazione alla evoluzione di un'altra azione musicale *M*. Data una istanza di *M*, la sua componente analogica è formata da un frammento di partitura ("agganciato" al concetto *M*) e da un insieme di funzioni, o metodi, agganciati alle sue relazioni, che estraggono le caratteristiche dal frammento di partitura. Una richiesta (o "query") può essere inoltrata al sistema ad esempio sulla intensità di M, corrispondente ad una chiamata al metodo *intensity* di *M*, operante sul frammento di partitura associato ad M. In tal modo, un processo di ragionamento è portato avanti parzialmente in termini procedurali. Si noti che non è stata effettuata alcuna assunzione a priori sulle caratteristiche essenziali delle azioni musicali: nella musica classica occidentale, l'aspetto melodico può essere preponderante, tuttavia il sistema consente di modificare, o ridefinire sia relazioni che i "manipolatori" secondo necessità peculiari.

Una *compositional action* è caratterizzata da una connotazione temporale (in un asse temporale differente dalle azioni musicali, definito dai ruoli *Mbegin* e *Mend*) e da una descrizione procedurale del suo comportamento. Così come per ogni oggetto definito nel sistema, il compositore può definire sia sottoclassi che istanze di azioni compositive.
compound compositional action è definita come un insieme di compositional action, la cui esecuzione è governata da opportuni vincoli sulle loro relazioni temporali (concorrenza, sincronizzazione).

E' possibile utilizzare differenti interfacce per comunicare con il sistema: in particolare, è disponibile una presentazione visuale del formalismo a reti semantiche. Il compositore interagisce quindi in modo grafico o testuale con questa parte del sistema introducendo e/o modificando elementi nella rete, nonchè specificando richieste, cui il sistema risponde in modo visuale, dove possibile. Richieste possono essere effettuate sia per quanto riguarda la parte terminologica della rete (classi e sottoclassi), che per quanto riguarda entità musicali individuali (istanze).
Possono essere visualizzati sottoinsiemi del materiale musicale complessivo, permettendo differenti presentazioni dello stesso materiale. Ad esempio, in figura 1, la richiesta "mostra tutti gli oggetti da cui *canon* eredita proprietà", fornisce come risultato una evidenziazione (parte in grigio) di una parte della rete.
Un'altra semplice richiesta potrebbe essere "mostrami tutto ciò che conosci del concetto *music action*, ad eccezione delle sue relazioni temporali": il risultato a questa "query" è mostrato in figura 1, box "a".

Da un punto di vista della intelligenza artificiale, la componente simbolica di HARP è equivalente ad un sottoinsieme della logica del prim'ordine. La risposta a "query" è trovata mediante l'apparato deduttivo di tale formalismo. Un *classificatore* [2] è stato implementato per organizzare concetti in tassonomie.

4. HARP: Il sottosistema analogico

Azioni musicale hanno "ganci" (puntatori) a frammenti di partiture, azioni compositive hanno invece "ganci" a frammenti di codice. Questo livello sottostante di definizioni procedurali costituisce il *livello analogico* di HARP, che è strutturato in una parte *sonologica* ed in una *simulativa*. La componente *simulativa* è formata da un insieme di procedure che agiscono come la controparte dell'apparato deduttivo simbolico ed hanno il ruolo di integrare le capacità deduttive di tale apparato. Per questa ragione, viene chiamato, in HARP, *simulative engine*.

Entrambi i componenti, sonologico e simulativo, sono guidati dal livello simbolico, dato che opportune attivazioni di un nodo nella rete semantica corrisponde a chiamate al codice corrispondente. Meccanismi per una interazione visuale sono presenti anche a livello analogico. Essi si basano su notazione musicale, descrizioni grafiche di suoni e di metafore (campi di forze, dinamica, come descritto più avanti). In questo ambito, "query" corrispondono ad attivazioni di moduli di codice, con opportuni parametri. Ad esempio, in figura 1 (box "b") si nota che un *canon* è *generated_by* un processo di imitazione (*imitation*). Una particolare sottoclasse di canon, ad esempio *per diminutionem canon*, può essere definita, come *generated_by* una sottoclasse di imitazione (*per diminutionem imitation*). Procedure più dettagliate possono essere agganciate a quest'ultimo concetto, ed una di esse può implementare un opportuno algoritmo di diminuzione. Una istanza di *per diminutionem canon* può così essere generata, a partire dai concetti sopra definiti e da una istanza di *antecedent*, inoltrando una "query" su un opportuno conseguente per questo canone. Il sistema risponde attivando le procedure agganciate a *per diminutionem imitation*, passando come parametro l'istanza di *antecedent*.

4.1 L'uso di metafore: i campi di forze

Un insieme significativo di descrizioni analogiche è quello basato sulla metafora dei *campi di forze*: ciò consente al compositore di pensare ed eseguire un insieme di azioni compositive in termini della intuitiva dinamica della navigazione in campi attrattivi e repulsivi, invece che in termini di regole. Ad esempio, variazioni timbriche nel continuo possono essere pensate come campi di forze operanti nello spazio timbrico, cui azioni musicali sono soggette in definiti intervalli di tempo.

"Query" analogiche complesse possono quindi essere formulate, in modo da coinvolgere attivazioni complesse di procedure interagenti, allo scopo di costruire azioni musicali complesse.

In breve, la metafora dei campi di forze sembra utile per (i) descrivere cambiamenti nel continuo di azioni musicali: la risultante di diversi campi di forze può modellare facilmente comportamenti complessi (come evoluzioni timbriche complesse), altrimenti difficilmante modellabili con regole; (ii) fornire un differente punto di vista del materiale musicale, così come differenti, più evocative primitive visuali di manipolazione.

Meccanismi di interazione visuale ed ipermediale possono essere definiti mediante metafore quali i campi di forze, ad esempio per controllare esecuzioni all'elaboratore, particolari parametri di partiture, verso una sorta di "partitura animata" o di opera multimediale [11]. Studi ed esperimenti in questa direzione sono in via di sviluppo.

5. Conclusioni

Diverse altre caratteristiche sono incluse nel sistema, come la possibilità di fondere diverse basi di conoscenza (una sorta di fusione di stili), mantenendo una consistenza globale e rilevando possibili conflitti. Altre caratteristiche sono in fase di implementazione. Problemi aperti ancora presenti nel formalismo, come la possibilità di memorizzare nel sistema diversi punti di vista dello stesso oggetto, costituiscono area di investigazione per una attività di ricerca futura.

Un brano musicale complesso, *Anceps Imago* di Corrado Canepa (1989/90), è stato utilizzato per testare le potenzialità del formalismo [4].

Il sistema HARP è attualmente in fase prototipale, implementato in Arity Prolog e Microsoft C in ambiente Windows SDK.
Il linguaggio utilizzato in HARP per la descrizione a basso livello di partiture e per la esecuzione è *cmusic* [1], in una versione per sistemi MS-DOS sviluppata presso il laboratorio di informatica musicale del DIST [9], a partire dalla versione per ambiente Unix implementata al CARL, Università di San Diego, California.

6. Ringraziamenti

Gli autori desiderano ringraziare gli studenti Attila Baldini e Ciro Lo Basso per il loro contributo alla implementazione della interfaccia grafica del prototipo software.
Il presente lavoro è supportato parzialmente grazie al *Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo* (LRC C4 MUSIC) del Consiglio Nazionale delle Ricerche (CNR) (contratto no. 90.00716.PF69, pos. 115.14231).

Riferimenti bibliografici

[1] G. Loy, and C. Abbott, "Programming Languages for Computer Music Synthesis, Performance and Composition", **ACM Computing Surveys**, Vol. 17, No. 2, June 1985, pp.235-265.

[2] R.J. Brachman, and J.G. Schmolze, "An overview of the KL-ONE knowledge representation system", **Cognitive Science**, Vol. 9, 1985, pp.171-216.

[3] A.Camurri, C.Canepa, M.Frixione, and R.Zaccaria, "HARP: A system for Intelligent Composer's Assistance", **IEEE Computer**, special issue on Computer Generated Music, Vol.24, N.7, July 1991, pp.64-67, IEEE Computer Society Press.

[4] A.Camurri, C.Canepa, M.Frixione and R.Zaccaria, "HARP: A framework and a system for Intelligent Composer's Assistance", In D.Baggi (Ed.), **Readings in Computer Generated Music**, 1991, IEEE Computer Society Press.

[5] A.Camurri, "Applications of Artificial Intelligence Methodologies and Tools for Music Description and Processing", In G. Haus (Editor), **Music Processing**, The Computer Music and Digital Audio Series, J. Strawn (series Editor), A-R Editions, Madison, Wisconsin, USA, 1991.

[6] A.Camurri, "On the Role of Artificial Intelligence In Music Research", **Interface**, 19(2-3), pp.219-248, 1990, Swetz&Zeitlinger, Lisse, The Netherlands.

[7] S.T.Pope, "Object-Oriented Design in the MODE", Int. Rep., 1990.

[8] W.Kim, and F.H.Lochovsky, **Object-Oriented Concepts, Databases, and Applications**, ACM Press, 1989.

[9] A.Camurri et al., "Note tecniche relative al modulo DIST.Music.Tool", Int. Rep. DIST, Università di Genova.

[10] M.Leman, "A process model for musical listening based on ΩH-networks", **CC-AI**, Vol.3, No.3, pagg.225-239.

[11] A.Camurri, C.Canepa, F.Orlich, and R.Zaccaria, "Interazioni Musica-Movimento: Un Sistema per la Generazione di Partiture da Animazioni 3D", Proc. **VII Colloquium on Musical Informatics**, Roma, 1988.

[12] SIGART Bulletin, Special Issue on Implemented Knowledge Representation and Reasoning Systems, Vol.2, No.3, June 1991. ACM Press.

SOUL: UN SENSORE ACUSTICO ADATTIVO PER IL RICONOSCIMENTO DI SORGENTI SONORE

U.Bertelli, C.Bima, A.Camurri, L.Cattaneo, P.Jacono, P.Podestà, R.Zaccaria

DIST - Universita' di Genova
Laboratorio di Informatica Musicale
Via Opera Pia 11/A - 16145 Genova
posta elettronica: music@dist.dist.unige.it

Abstract

One of the main goals of intelligent musical signal processing is building systems that *listen to* and understand music by relating it to previously stored knowledge, either in a symbolic or subsymbolic approach. "Artificial intelligent listeners" should accomplish a signal-to-symbol transformation.

The SOUL (Self-Organizing Universal Listener) project aims at defining a model and develop a prototypical system of an intelligent adaptive acoustic sensoring device, based on both digital signal processing techniques and recent neural approaches.

SOUL is motivated essentially by two different domains of application: sound and music processing, and acoustic sensoring applications in advanced robotics.

From the point of view of sound and music processing, the project reveals a significant interest. The SOUL project configures itself as a module of a wider project of a workstation for sound and music processing. We are currently designing its integration with the HARP system, described elsewhere in these proceedings: the goal is to develop a global system in which the objects manipulated by the system, representing "chunks" of musical knowledge, can be derived both by explicit (symbolic) user's definitions or queries to the system, and by exposing the system to music sources, basing on the SOUL signal-to-symbol transformation model. See [5] in these proceedings and [6] for a deeper discussion on the definition of symbolic musical properties emerging from a subsymbolic layer.

In the field of mobile robots, that is intelligent autonomous systems operating in (partially) unknown and time-varying environments, there is a strong need of acoustic sensoring systems. No significant results are currently available from traditional signal processing techniques. Recent developments of both hardware (parallel architectures) and modeling architectures (i.e. neural networks) should allow to solve this class of problems.

SOUL is concerned with the modeling of the psycho-acoustic skills of a human listener, that is the capability of recognizing music sources (instruments) from a complex input (the overall music input, for example formed by a group of different instruments).

A basic, simple consideration which has influenced the design of the system is that most of human sound recognition processes conserve a good performance even in case of considerably reduced bandwidth and quality of the signal (noise). A piece of music heard from an AM radio station has a

bandwidth of about 4KHz: even in this case, the human ear often reaches "good" performances, in the sense that it is able, for example, in most cases, to recognize and follow particular instruments. This consideration led us to a design of a part of the ear model computationally simpler and more efficient.

The architecture of the current prototypical system is shown in figure 1. It is based on the main components described below. Three sound processing modules, devoted to the acquisition and pre-processing of complex real sound signals, enhance the quality of the incoming sound signal and then process the resulted signal basing on models of the human ear. The output obtained from these modules is processed by a neural architecture based on the model proposed by Kohonen [1]: this module has the goal of classifying and extracting particular sound sources.

These modules should require an implementation on special purpose hardware for both real-time digital signal processing capabilities and efficient neural networks implementations: however, only a software implementation has been developed at now.

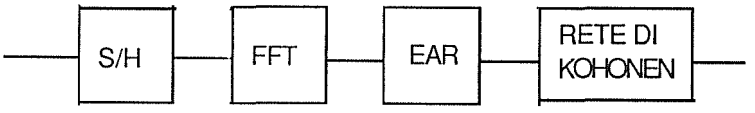


Figura 1: SOUL: Architettura generale

1. Introduzione

In ambito musicale e psico-acustico, è nota la capacità biomorfa di estrarre la singola sorgente sonora (lo strumento) da un background non necessariamente definibile in termini di grandezze tipiche della teoria dei segnali (frequenze, intensità, correlazione, ecc.). Si tratta di una attività intelligente, in quanto coinvolge livelli di astrazione, e adattiva, basata sull'apprendimento.

La tecnologia del sensore periferico è già ben consolidata e disponibile a basso costo (il trasduttore elettroacustico), mentre è da investigare la parte intelligente del sistema sensoriale, per cui la tecnologia corrente (elaborazione digitale su h/w dedicato, sistemi basati su conoscenza, data fusion) non è soddisfacente.

L'articolo descrive brevemente il modello complessivo ed una realizzazione software prototipale di un sensore acustico intelligente adattivo per il riconoscimento e l'inseguimento di una sorgente sonora. La ricerca descritta nell'articolo si basa sull'impiego di reti neurali. Il modello che è stato scelto è quello delle *self-organizing networks*, sviluppato da Kohonen [1]. Tale tipo di reti neurali consente, dopo opportuno training, la formazione di aggregazioni di unità accordate a particolari pattern.

Il modulo prototipale SOUL (*Self-Organizing Universal Listener*) consiste di un sistema di simulazione integrato per la acquisizione ed il pre-processing del suono, unito ad una interfaccia grafica per il monitoraggio della rete neurale. L'articolo descrive brevemente l'architettura del

sistema e fornisce alcuni dati sulle prestazioni relativamente al riconoscimento di particolari strumenti (sax tenore e contrabbasso).

2. Architettura generale

In figura 1 è mostrata l'architettura del sistema, composto di quattro moduli: un campionatore (sample and hold), un modulo per la trasformazione in frequenza (fast fourier transform), un modulo per la estrazione dei parametri caratteristici del segnale (estrazione di fondamentali e rapporti armonici, basato su di un modello approssimato dell'orecchio umano), infine un modulo per la classificazione del segnale basato su reti di Kohonen.

Un segnale di ingresso s(t) continuo nel tempo viene campionato con un modulo di sample/hold, creando un file di campioni nel tempo. Per un'analisi del segnale in frequenza, è stato sviluppato un modulo per la FFT del segnale (Fast Fourier Transform). Una volta ottenuto lo spettro del segnale, è necessario farne un'analisi che evidenzi le caratteristiche intrinseche dello strumento. E' questa la fase più delicata. Infatti non esiste, al momento, una metodologia consolidata che permetta di estrarre, dallo spettro di un segnale complesso, le caratteristiche di una data sorgente sonora. Ciò che si è tentato di fare è di modellare l'orecchio umano in modo da sfruttare al massimo l'informazione contenuta negli spettri del segnale campionato, per ricavare quelle differenze atte al riconoscimento dello strumento. Questo modulo è in grado, nei casi sperimentati, di estrarre il contributo, in modulo, della frequenza fondamentale e delle successive armoniche della nota suonata. Questo insieme di informazioni costituisce l'ingresso ad una rete neurale di Kohonen bidimensionale, in grado di classificare sia il tipo di strumento che la nota suonata. I moduli fondamentali sono raffigurati in figura 1.

3. Modello dell'orecchio

L'analisi in frequenza viene attuata da un array di sensori nella coclea dell'orecchio. La frequenza caratteristica di questi sensori dipende dalla loro locazione lungo la coclea e cresce in maniera monotona lungo l'array. Ogni sensore si comporta come un filtro passabanda con una larghezza di banda uguale a circa il 10% della frequenza caratteristica. Inizialmente si è pensato di coprire una banda di 20 Khz come è quella dell'orecchio umano. Inoltre, si sono sposate le indicazioni descritte in [4], che suggerisce un filtraggio a tre livelli: al primo livello ciascun sensore è sensibile ad un intervallo centrato nella frequenza caratteristica di ampiezza +/-5% della frequenza caratteristica; inoltre, tra due sensori successivi esiste una differenza di frequenza dello 0.02%. Considerando un range di otto ottave, il numero totale di sensori risulta essere di circa 28000. Le uscite dei sensori vengono date in input ai JND (*Just Noticeable Difference*) BUCKETS in modo tale che la differenza fra due neuroni successivi sia del 0.3%; ogni neurone di questo livello ha come ingresso 19 uscite dei sensori del livello inferiore. Infine, ogni sensore dell'ultimo livello rappresenta un semitono; la differenza in frequenza fra due semitoni è del 6% (figura 2).

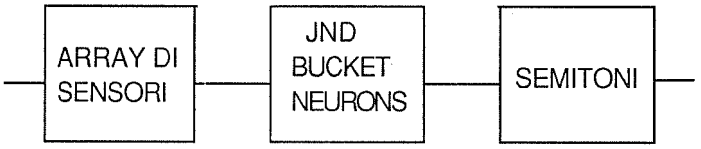


Figura 2: Filtro per l'estrazione dei semitoni

Questo approccio, in effetti, presenta lo svantaggio di un numero spropositato di neuroni da allocare in memoria. Si è quindi stabilito che la banda da coprire non superasse i 4Khz (circa 5 ottave, quindi 60 semitoni invece dei 96 nel caso di 8 ottave), limite sufficiente per garantire ancora una buona capacità di riconoscimento da parte dell'orecchio umano (è la stessa banda di una stazione radio in AM); inoltre, si è deciso di ridurre a 2 i livelli di pre-processing dei campioni in frequenza. Si è quindi definito un primo algoritmo, descritto dai passi seguenti:

- calcolo della frequenza corrispondente a ciascun semitono;
- tramite la formula $K(i+1)/K(i)=1.003$, calcolare la frequenza caratteristica dei neuroni del primo livello;
- calcolare il valore assunto da ciascun neurone del primo livello;
- calcolare il valore assunto da ciascun semitono;
- individuare la fondamentale e le prime tre armoniche;
- calcolare tutti i possibili rapporti tra le armoniche che vanno a costituire un vettore di ingresso per la rete di Kohonen.

Gli ingressi ad un neurone del livello successivo vengono pesati attraverso una gaussiana centrata sulla frequenza caratteristica del neurone stesso.

3.1 Il riconoscitore neurale

Un modello di rete neurale innovativo in diversi suoi aspetti è quello proposto da Kohonen in [1], e le sue recenti estensioni. Tale modello innanzitutto stabilisce una struttura della rete su un unico livello di neuroni, aventi tutti gli stessi ingressi (quelli della rete). Altra caratteristica specifica è che ciascun neurone presenta delle connessioni con il resto dei neuroni che compongono la rete, connessioni dette *lateral feedback*. I pesi di tali connessioni variano secondo una funzione *DOG* (figura 3), centrata sul singolo neurone. Con un processo di adattamento agli ingressi, sulla rete si vengono a costituire delle zone (*bubble*) più o meno sensibili a ciascun ingresso durante la fase di training. Il centro della singola *bubble* rappresenta l'uscita corrispondente ad un ingresso fissato: si tratta cioè di una risposta localizzata. Si può parlare, quindi, di topologia e di dimensione della rete. L'algoritmo di Kohonen produce un mapping da uno spazio n-dimensionale ad uno bidimensionale (nel nostro caso specifico), la griglia dei neuroni. Associato ad ogni neurone esiste un insieme di sinapsi e un set di neuroni pre-sinaptici che costituiscono un *input-layer*. Inizialmente tutte le sinapsi hanno pesi casuali. Le sinapsi vengono adattate secondo la migliore corrispondenza tra il vettore di ingresso e le sinapsi dei singoli neuroni. I pesi sinaptici del neurone che dà luogo alla migliore corrispondenza e quelli dei neuroni topologicamente vicini vengono leggermente cambiati in modo che la somiglianza fra i vettori dei pesi delle sinapsi ed il vettore degli ingressi cresca.

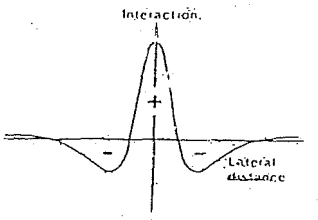


Figura 3: La funzione DOG

L'algoritmo può essere riassunto come segue:

```
1) Inizializzare i pesi sinaptici dando loro valori casuali;
2) Inizializzare il tasso di apprendimento e il raggio della bubble;
3) Randomizzare i dati di input;
4) Calcolare la distanza euclidea tra il vettore di input e il vettore sinaptico;
5) Sceglere il neurone con il vettore sinaptico più simile a quello di ingresso;

6) Per tutti i neuroni che cadono nel raggio della bubble centrata sul neurone
   fissato, adattare le sinapsi secondo la formula:
                          w(t+1)=w(t)+a(t)*[v(t)-w(t)]
   dove:
      w(t+1)  è il vettore sinaptico all'istante t+1;
      w(t)  è il vettore sinaptico all'istante t;
      a(t)  è il tasso di apprendimento all'istante t;
      v(t)  è il vettore di input al tempo t;
7) Decrescere il tasso di apprendimento ed il raggio della bubble, se necessario;
8) Tornare al passo 3).
```

4. Osservazioni sulle prove effettuate

Abbiamo utilizzato il software prototipale inizialmente per riconoscere il contrabbasso in un segnale disturbato. La scelta dello strumento non è casuale, infatti il contrabbasso presenta caratteristiche timbriche che lo rendono di più facile riconoscimento. Avevamo a disposizione una registrazione analogica di due scale di note suonate con tale strumento. Ogni nota è stata isolata e il segnale corrispondente è stato utilizzato come ingresso alla cascata di moduli di figura 1.

Sono state utilizzate diverse reti di Kohonen, differenti fra loro per dimensioni, durata del ciclo di apprendimento e dimensione del vettore di ingresso. Alle reti così allenate, sono stati presentati vettori di test per verificare se realmente fossero in grado di riconoscere la presenza del contrabbasso. I risultati ottenuti in questo primo esperimento sono stati poco soddisfacenti. Le cause possono essere ricercate essenzialmente in (i) una cattiva qualità della registrazione analogica (il segnale include rumore); (ii) il test di riconoscimento è stato effettuato per un contrabbasso differente da quello usato per addestrare la rete.

Vediamo ora le differenze principali tra i vari tipi di reti di Kohonen provate. Si è notato il diverso comportamento della rete al variare della dimensione del vettore d'ingresso: più questa aumenta più i semitoni si sparpagliano in essa; ciò perchè vengono evidenziate le diverse

caratteristiche delle note stesse in quanto si ha a disposizione una quantità di informazione maggiore. Quando si considera il vettore di ingresso di grandezza 15 lavoriamo con i rapporti tra le prime 6 armoniche, mentre nel caso del vettore di ingresso pari a 4 abbiamo solo i rapporti tra le armoniche 2, 3, 4 con la fondamentale.

In questo prototipo, in cui il compito è solo di riconoscere se suona oppure no il contrabbasso, la distribuzione dei semitoni sulla rete è uno svantaggio. Infatti vorremmo che i semitoni si concentrassero il più possibile in una zona ben definita della rete, quasi da creare un'area che potremmo definire "area del contrabbasso"; in questo modo, se durante la fase di riconoscimento viene eccitato un neurone al di fuori della zona, si sarebbe sicuri di stare trattando un file di campioni in cui non è presente il contrabbasso.

Purtroppo quello detto sopra è puramente teorico in quanto è necessario trovare un compromesso tra la dimensione del vettore di ingresso e la comunque inevitabile distribuzione dei semitoni.

Altro punto da far notare è la durata del ciclo di apprendimento. Abbiamo rilevato che aumentando la durata, portandola da 100 a 500, non si ottengono risultati migliori, anzi peggiorano. Questo è dovuto al fatto che, anche in questo caso come in quello del vettore di ingresso, più si prolunga il ciclo più vengono amplificate le differenze fra i semitoni, per cui risulta più difficile riconoscere il suono del contrabbasso contaminato da altri strumenti o dal rumore.
Per quanto riguarda la dimensione della rete, in questo esperimento non abbiamo ricavato risultati tali da far preferire inequivocabilmente un tipo di rete rispetto alle altre.

Un ulteriore serie di esperimenti è stata quindi effettuata, cercando di riconoscere un sax tenore. La metodologia usata è identica a quella per il contrabbasso. I risultati ottenuti sono, però, significativamente migliori. Infine abbiamo provato a riconoscere il sax tenore utilizzando una rete addestrata per il contrabbasso. Ciò che abbiamo ottenuto è sorprendente. Infatti in questo caso viene identificata una zona precisa della rete come "zona sax tenore" in modo indipendente da quale sia la nota suonata.

5. Conclusioni e sviluppi futuri

E' attualmente in corso la definizione di un modello più evoluto del modulo per l'estrazione di parametri dal segnale, da fornire come ingresso alla rete di Kohonen. Esso si basa sull'algoritmo di Terhardt [2, 3] e sulle recenti estensioni di Parncutt. Il modello descritto della coclea consente, mediante il filtraggio del segnale con il banco di filtri a Q costante, di riconoscere anche sorgenti con fondamentale di ampiezza inferiore alle armoniche. Gli algoritmi definiti da Terhardt e Parncutt consentono, fra l'altro, il riconoscimento di sorgenti sonore anche nel caso in cui la frequenza fondamentale ha intensità molto inferiore alle armoniche o addirittura è mancante. I risultati ottenuti utilizzando questo più raffinato modello saranno descritti in un prossimo articolo.

Ringraziamenti

Desideriamo ringraziare Marc Leman per i preziosi suggerimenti e collaborazione nell'ambito del progetto SOUL.

Il presente lavoro è supportato parzialmente dal *Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo* del Consiglio Nazionale delle Ricerche (CNR), linea di ricerca coordinata C4:MUSIC (contratto n. 90.00716.PF69, pos. 115.14231).

Riferimenti bibliografici

[1] T. Kohonen, **Self-Organization and Associative Memory**, Springer-Verlag, 1988.

[2] E. Terhardt, G. Stoll and M. Seewann, "Pitch of complex signals according to virtual-pitch theory: Tests, examples, and predictions", **J. Acoust. Soc. Am.**, Vol.71, No.3, March 1982.

[3] E. Terhardt, G. Stoll and M. Seewann, "Algorithm for extraction of pitch and pitch salience from complex tonal signals", **J. Acoust. Soc. Am.**, Vol.71, No.3, March 1982.

[4] K. Lent, "An Efficient Method for Pitch Shifting Digitally Sampled Sounds", **Computer Music Journal**, Vol.13, No.4, 1990, MIT Press.

[5] M. Leman, "Tonal context by pattern-integration over time", Proc. **IX Colloquium on Musical Informatics**, Genova, 1991 (this volume).

[6] M.Leman, "Symbolic and Sub-Symbolic Description of Music", in G.Haus (Ed.), **Music Processing**, 1991, A-R Editions.

[7] A.Camurri, M.Capocaccia, and R.Zaccaria, "Experimental Neural Accompanist (ENA)", Proc. **International Neural Networks Conference, INNC-90**, Paris, Kluwer Academic Publ.

Capitolo 4: Sistemi per l'Informatica Musicale

The IRCAM Musical Workstation: A Prototyping and Production Tool for Real-Time Computer Music

Cort Lippe, Zack Settel, Miller Puckette and Eric Lindemann
IRCAM, 31 Rue St. Merri, 75004 Paris, France

During the past three years IRCAM has developed the IRCAM Musical Workstation (IMW). Recently we have begun work on musical tools and the realization of music with the system. Miller Puckette has written a version of MAX for the IMW which includes signal processing objects, as well as the standard objects found in the Macintosh version of MAX (including MIDI objects). The MAX environment for the IMW offers composers and researchers a powerful real-time prototyping and production tool for signal processing, synthesis, and the detection of continuous control parameters from musical instruments.

The IRCAM Musical Workstation (IMW) [1] is built around a dual Intel i860 card developed at IRCAM [2], and commercialized by ARIEL Corporation, which plugs into a NeXT machine cube. The IMW attempts to break with the paradigm wherein a DSP is controlled by a separate host computer. The i860 is a general-purpose RISC processor, in other words, general enough and yet fast enough to do control and synthesis at the same time.

MAX

MAX [3], as developed by Miller Puckette, extended by David Zicarelli, and commercialized by Opcode, is a graphical programming environment for the Macintosh with MIDI handles. MAX now runs on the IMW with an additional library of signal processing objects. So far, Puckette has written about 30 signal processing objects, including objects for filtering, sampling, pitch tracking, delay lines, FFTs, etc. MAX runs in the FTS [4] real-time monitor program under the CPOS operating system [5].

With the IMW version of MAX, the flexibility with which one can create control patches in the original Macintosh version of MAX is carried over into the domain of signal processing. Creating an oscillator object and a DAC object, hooking them together, and controlling the oscillator's frequency and amplitude via a MIDI keyboard in real-time becomes an easy task. (Figure 1.)

Prototyping Environment

In musical applications the ability to test and develop ideas interactively plays an important role. Because of its single architecture the IMW is a powerful prototyping and production environment for musical composition. Prototyping in a computer music environment often combines musical elements which traditionally have fallen into the categories of "orchestra" (sound generation) or "score" (control of sound generators). Mainly due to computational limitations, real-time computer music environments traditionally have placed hardware boundaries between "orchestra" and "score": the sound generation is done on one machine while the control is done remotely from another. The 4X machine at IRCAM and the MIDI studio are classic

examples of this dichotomy. When developing a synthesis algorithm which makes extensive use of real-time control, it is extremely helpful, if not essential to develop the synthesis algorithm and the control software together. This is greatly facilitated when both the sound generation and control run on the same machine and in the same environment. Using MAX on the IMW, both synthesis and control can be developed and fine-tuned simultaneously in real time on the same machine with straightforward bi-directional communication between the two domains (without, for example, having to format communications protocol for a hardware link such as MIDI or RS232). (Figure 2.) A synthesis algorithm and its control algorithm may be grouped and saved together, forming a single complex object which can be utilized in other configurations (patches), ultimately allowing the user to blur the boundaries of the traditional computer music orchestra/score paradigm.

Production Environment

Musical production has just begun on the IMW. Outside of several sketches (Lippe and Settel) our early experiences have consisted of porting compositions made at IRCAM using the 4X to the IMW. This has meant porting the 4X micro-code, the C code from the 68000 host machine which controls the 4X, and finally MAX control programs which run on a Macintosh and communicate with the 4X host via MIDI. Reduction of all this code from a network of three machines to a single machine and a single environment on the IMW has been quite successful. For example, one module that was coded across the three machines which allows for independence of pitch and time in playback of sampled sounds was reduced to a single rather simple MAX patch upon being transferred to the IMW.

A typical 4X piece making use of a variety of modules for signal processing and sound synthesis might contain the following: harmonizers, delay lines, frequency shifters, samplers (with recording and playback in real-time), one or two synthesis algorithms such as additive synthesis or *Phase Aligned Formant Synthesis* (developed by Puckette) , filtering, reverberation, spatialisation, and possibly an FFT (for the analysis of incoming signals in order to extract control information for additive synthesis). Finally, a crossbar enabling all signals to be sent from any module to any other is normally included to maximize the number of possible signal paths. A configuration of this complexity for the composition *Pluton* [6] by Philippe Manoury (a 50 minute piano/4X piece) has been ported to the IMW.

Current Developments

Real-time signal analysis of instruments for the extraction of musical parameters gives composers useful information about what an instrument is doing. One of the signal processing objects recently developed in MAX on the IMW is a pitch tracking algorithm. In a clarinet sketch produced by Lippe, the pitch detection algorithm analyzes the incoming clarinet signal and outputs MIDI-style pitches which are sent to a score follower using the Explode object [7] which in turn triggers and controls the electronic events in the score. The pitch follower also outputs continuous control information analogous to pitch-bend in the MIDI domain. (All of this is done without the use of a MIDI cable, and thus without the bandwidth limits of MIDI.) This information coupled with envelope following can act as a secondary aid to score following. Certain kinds of musical material are very difficult to recognize with pitch following alone: thus one can rely on other analyses, depending on the context, to augment pitch-oriented score following, or follow other aspects of an instrument's signal when pitch is irrelevant or difficult to detect.

A subject of great interest at IRCAM presently is real-time audio signal analysis of acoustic instruments for the extraction of musical parameters. Musical parameters which can be derived from continuous tracking include a rich variety of elements. In the frequency domain, using pitch tracking, one can determine the stability of pitch on a continuous basis for the extraction of pitch-bend, portamento, glissando, trill, tremolo, etc. In the amplitude domain, envelope following of the continuous dynamic envelope for articulation detection enables one to determine flutter-tongue, staccato, legato, sforzando, crescendo, etc. In the spectral domain, FFT, pitch tracking, and filtering can be used to track continuous changes in the spectral content of sounds allowing for detection of multiphonics, inharmonic/harmonic ratios, timbral brightness, etc. High-level event detection combining the above mentioned analyses of frequency, amplitude, and spectral domains can provide rich control signals that reflect subtle changes found in the input signal. These control signals carry musically expressive information which can be used to drive signal processing and sound generating modules, ultimately providing an instrumentalist with a high degree of expressive control over an electronic score. In a clarinet sketch by Settel, high level event detection plays an important role in the soloist's control interface to signal processing and synthesis modules. (Figure 3.)

The dynamic relationship between performer and musical material, as expressed in the musical interpretation, can become an important aspect of the man/machine interface for the composer and performer (as well as the listener) in an environment where musical expression is used to control an electronic score. The richness of compositional information useful to the composer is obvious in this domain, but other important aspects exist: compositions can be fine-tuned to individual performing characteristics of different musicians, intimacy between performer and machine can become a factor, and performers can readily sense consequences of their performance and their musical interpretation.

Future Developments

Future developments in MAX on the IMW include the implementation of a complex vocoder (Figure 4.), real-time direct-to-disk soundfile recording and playback, as well as interfaces developed in ANIMAL [8], an interface prototyping environment designed to control MAX patches. An ANIMAL interface is in the planning stages which will offer a toolbox for high-level manipulation of continuous control parameters and allow composers to fine-tune signal analysis of instruments dependent on their acoustical properties and the musical context.

References

[1] Eric Lindemann *et al,* "The IRCAM Musical Workstation: Hardware Overview and Signal Processing Features", Proceedings, ICMC 1990 (Glasgow).

[2] Cort Lippe, "Musical Performance Using the IRCAM Workstation", Proceedings, ICMC 1991 (Montreal).

[3] Miller Puckette, "The Patcher", Proceedings, ICMC 1988 (Cologne).

[4] Miller Puckette, "A Real-time Monitor for Multiprocessor Music Synthesis", To appear in the Computer Music Journal 15(3)

[5] Eric Viara, "A Real-time Operating System for Computer Music", Proceedings, ICMC 1990 (Glasgow).

[6] Cort Lippe, "A Technical Description of *Pluton* by Philippe Manoury" IRCAM Annual Report 1988, IRCAM, Paris

[7] Miller Puckette, "EXPLODE: A User Interface for Sequencing and Score Following", Proceedings, ICMC 1990 (Glasgow).

[8] Eric Lindemann, "ANIMAL: A Rapid Prototyping Environment for Computer Music Systems", Proceedings, ICMC 1990 (Glasgow).

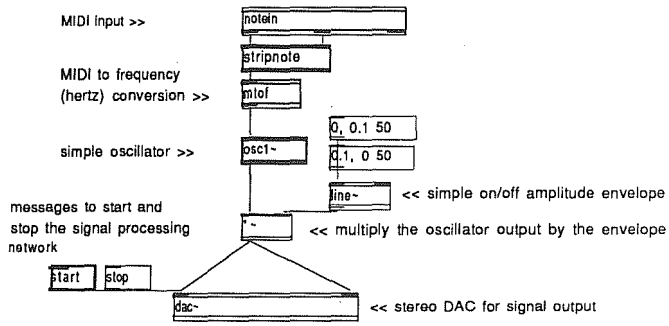


figure 1.

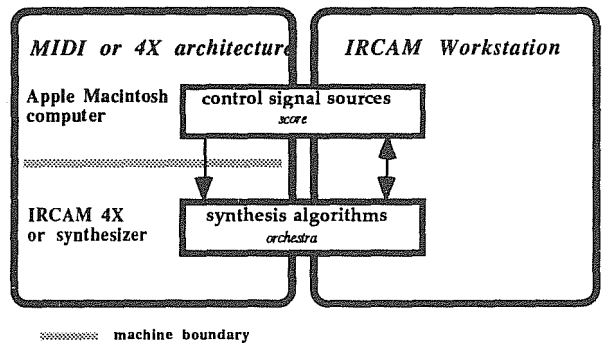


figure 2.

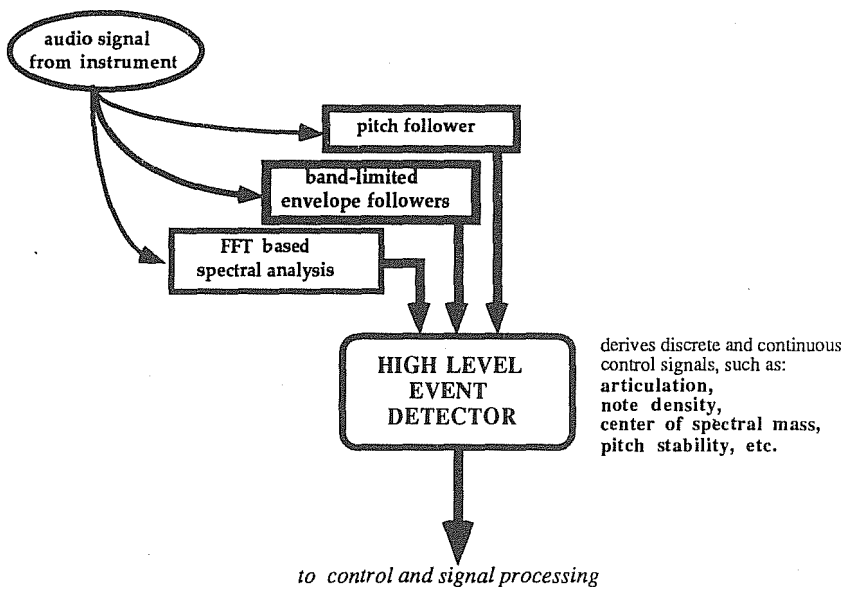


figure 3.

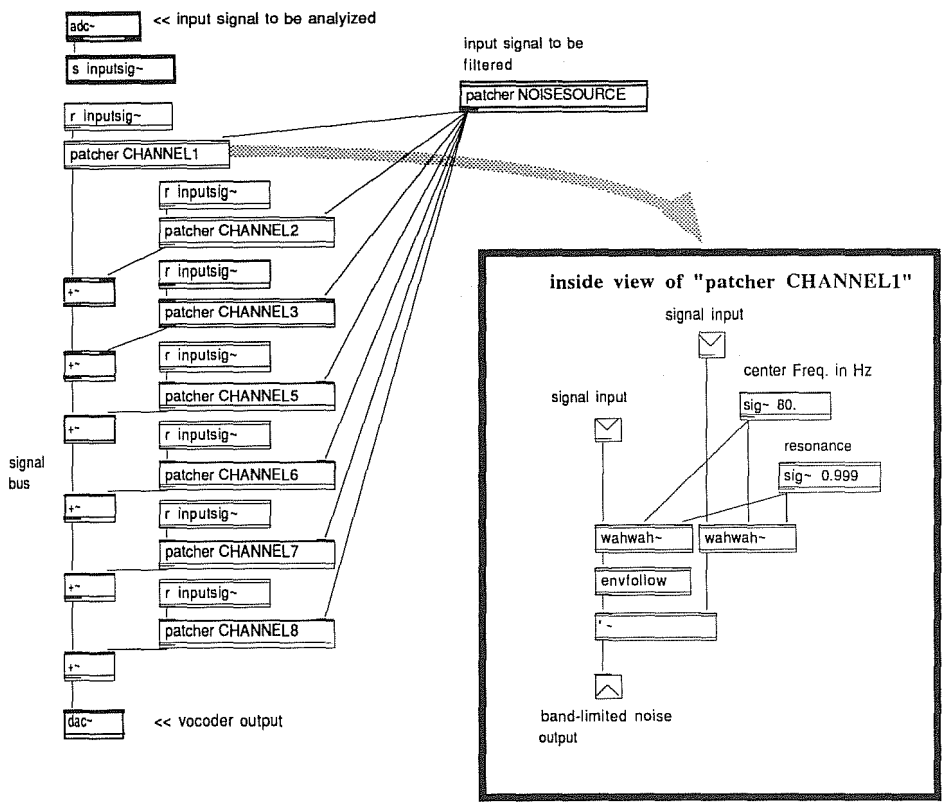

adc~
<< input signal to be analyzed
s inputsig~
r inputsig~
patcher CHANNEL1
input signal to be filtered
patcher NOISESOURCE
r inputsig~
patcher CHANNEL2
r inputsig~
patcher CHANNEL3
r inputsig~
patcher CHANNEL5
r inputsig~
patcher CHANNEL6
r inputsig~
patcher CHANNEL7
r inputsig~
patcher CHANNEL8
+~
signal bus
dac~
<< vocoder output
inside view of "patcher CHANNEL1"
signal input
center Freq. in Hz
signal input
sig~ 80.
resonance
sig~ 0.999
wahwah~
wahwah~
envfollow
band-limited noise output

figure 4.

INFORMATICA MUSICALE E NON-VEDENTI: UNA STAZIONE DI LAVORO SU PERSONAL COMPUTER

L. Camilleri*, F. Giomi*, P. Graziani**, L. Taggi**

*Divisione Musicologica CNUCE/C.N.R.
Conservatorio di Musica "L.Cherubini" di Firenze
P.zza delle Belle Arti 2, 50122 Firenze, Italy
E-mail: CONSERVA@IFIIDG.Bitnet
Tel: +39-55-282105

**Istituto di Ricerca sulle Onde Elettromagnetiche
del C.N.R.
Via Panciatichi 64, 50127 Firenze, Italy
Tel: +39-55-4378512

Keywords: Technology and Blindness, Handicap and Music, Computer Music Workstation, Vocal Synthesizer.

ABSTRACT

The research centers and fields which carry out research on technology and handicap are now numerous. These studies have the goal to supply to visually impaired, or to other handicapped, a possibility to help them in the every-day life and work by using new technologies.
The present project is a part of the Concerted Action on Technology and Blindness, a research program sponsored by EEC, and it is coordinated by the I.R.O.E. (institute of the National Research Council) with the collaboration of the Musicological Division of CNUCE/C.N.R. and the Conservatory of Music "L. Cherubini" in Florence.

In the field of music the new technologies can be of great support for the visually impaired. The main applications are oriented in the following guidelines:

1) Braille transcription of ink printed music:
It deals with computer programs which help the user to codify (with an alphanumeric code) the score and then to print them by means of a braille-printer.

2) Automated printing of musical score:
In this sector the research try to adapt or create new programs for printing music in conventional notation. In fact the commercial software use graphic user interfaces which make these systems unusable for blinds. It will allow to the blind musician to prepare and print a score in conventional notation without the need of help of a copyst or another person.

3) Musical education:
The new technologies make the access to musical education easier also for blind people. By means of new interfaces it is possible to make the learning of music easy and skip the problems.

4) Use of commercial programs by means of a vocal synthesizer:
It is possible to interface the commercial programs which do not use graphics by means of vocal synthesizer reading all the commands listed on the screen. In addition the user can hear the data inputted from the computer keyboard and control all the process. One example is the use of SCORE, a printing program which use alphanumeric input and text commands, by the help of this kind of device.

The project of a workstation usable also by visually impaired musicians concerns with a software allowing the encoding, processing, composition and managment of musical data.
The starting point of this system is the Teletau, a software package mainframe-based which permits several facilities of musical data managment whose user interface is completely text-oriented. This software comprehend facilities concerning (1) music encoding, (2) algorhitmic composition, (3) musical data processing, (3) handling of musical piece encoded library, (4) musical education, (5) music analysis. Sound output is provided by a RS232 Midi interface.
The current project concerns with the adaptation of the system in the MS-DOS environment. At present time the system can perform several operation of TELETAU system and codifies and manages musical pieces encoded by the Teletau own code system. By a link with Adagio software the sound output is provided using a standard Midi interface. Blind musicians can operate with this system using a vocal synthesizer that help them in the interaction, all made by text commands, with the software.

1. Introduzione

Sono ormai numerose le aree e i centri di ricerca che si occupano delle relazioni tra handicap e tecnologia. Gli studi a riguardo sono finalizzati sia alla possibilità di fornire ai non vedenti una formazione serie e rigorosa che permetta loro di avvicinarsi alle professioni dell'informatica sia alla creazione di una serie di ausili tecnici e applicazioni specifiche che ne agevolino il dialogo con gli elaboratori e ne consentano l'approccio ad attività professionali e culturali di qualsiasi tipo. Sono già esistenti in Europa alcune banche dati per la catalogazione degli strumenti oggi a disposizione, sia per quelli di uso personale che per quelli rivolti ai centri servizi (1).

Il nostro specifico progetto prende vita nell'ambito della *Concerted Action on Technology and Blindness*, un programma di ricerca promosso dalla COMUNITA' ECONOMICA EUROPEA. E' coordinato presso l'Istituto di Ricerca sulle Onde Elettromagnetiche del C.N.R. di Firenze con la collaborazione della Divisione Musicologica del CNUCE/C.N.R. e del Conservatorio di Musica di Firenze.

2. Informatica musicale e non vedenti

In questi ultimi anni anche l'informatica musicale, soprattutto con alcuni dei suoi numerosi settori, è divenuta patrimonio comune di musicisti e ricercatori non-vedenti, che hanno ottenuto così la possibilità di operare nei vari settori dell' elaborazione musicale e delle discipline musicologico-computazionali senza particolari difficoltà.

A questo proposito, è possibile delineare alcune linee guida verso le quali si orientano le principali applicazioni dell'informatica alla musica rivolte a utenti non vedenti:

1) TRASCRIZIONE BRAILLE DI MUSICA STAMPATA.

Si tratta di metodi con assistenza del calcolatore che sono basati su una procedura di input (mediante tastiera alfanumerica e mouse) nella quale l'utente deve soltanto riprodurre sullo schermo la partitura codificata. Si procede poi ad una trascrizione automatica in notazione Braille stampata.

Si muovono in questo senso diversi progetti europei. Vale la pena di ricordare quello sviluppato al centro TOBIA dell'Università di Tolosa da N. Baptiste e P. Sabatier; il sistema prevede alcuni moduli sequenziali: un editor grafico, un editor testuale, uno o più analizzatori, un generatore di file non formattati Braille ed infine un editor Braille. Dopo quest'ultimo modulo l'eventuale partitura è pronta per la stampa.

Al centro A.S.P.H.I. di Bologna, P. Bianco ha portato avanti un progetto analogo mentre M. Glover del Royal National Institute for the Blind di Londra sta lavorando su un sistema per la trascrizione in Braille partendo dalla codifica alfanumerica disponibile con il programma Score™.

2) PRODUZIONE AUTONOMA DI MUSICA STAMPATA.

Generalmente i sistemi commerciali disponibili per la composizione e l'editoria musicale non sono direttamente accessibili da persone non-vedenti a causa delle rappresentazioni grafiche e della forte interattività visiva sui quali sono basati. Si ha pertanto la necessità di sviluppare nuovi programmi "dedicati", o quantomeno speciali interfacce software per utilizzare quelli esistenti in modo da produrre composizioni originali e stampa di qualità.

Sono stati iniziati progetti di questo tipo sia in Italia che in Olanda.

3) DIDATTICA MUSICALE PER NON VEDENTI.

L'apprendimento della musica diventa accessibile a tutti grazie alle tecnologie informatiche e informatico-musicali. Anche in questo caso programmi specifici o interfaccie con quelli commerciali possono essere progettati.

Studi riguardo ad un lavoro specifico di interfacciamento uomo-macchina basato sulla tastiera musicale (per l'input dei dati) ed a un altro per la scrittura e l'ascolto musicale sono stati effettuati da M. Gabrieli del Conservatorio di Musica di Pescara.

Un altro esempio è ancora il sistema del centro TOBIA di Tolosa che prevede anche una apposita sezione per l'insegnamento dell'armonia basata su una base di dati, un insieme di regole ed una interfaccia di tipo colloquiale.

4) UTILIZZO DI PROGRAMMI COMMERCIALI CON SINTETIZZATORE VOCALE.

Si è in grado di interfacciare i normali programmi (non grafici) con un sistema di lettura dello schermo di personal computer MS-DOS: l'utente non vedente ascolta i segnali audio in lingua italiana emessi da un apposito sintetizzatore vocale (una descrizione più dettagliata verrà fornita più avanti).

Questo supporto software e hardware trova una sua buona applicazione in combinazione con il programma di codifica e stampa di partiture musicali Score™, progettato da Leland Smith della Stanford University e disponibile su Personal Computer MS-DOS. In questo caso le informazioni visive sono gestite da una speciale procedura che intercetta le richieste di "switching" grafico e le inibisce, in modo da fornire ogni informazione in modo testo.

3. La stazione di lavoro su Personal Computer

Le lacune che il nostro lavoro intende colmare sono quelle riguardanti software specifici per la codifica, elaborazione, composizione e gestione di brani musicali; si pone come

prioritaria l'esigenza di progettare e realizzare una stazione di lavoro che possa essere impiegata, anche da musicisti non-vedenti, con metodologia operativa relativamente semplice.

Proprio riguardo a tale metodologia, il sistema prende vita da un sistema per la "Computer Music" già ampiamente collaudato presso il Conservatorio di Musica di Firenze e costruito dalla Divisione Musicologica del CNUCE/C.N.R.: il *software package* Teletau.

Questo sistema è stato implementato su un calcolatore di tipo "mainframe" (IBM 3090) dotato di notevoli caratteristiche di memoria e velocità di elaborazione. E' previsto sia un accesso locale, con lavoro interattivo, che uno remoto attraverso la rete geografica euro-americana EARN-BITNET.

Le applicazioni sviluppabili con questo software possono essere raggruppate nelle seguenti categorie:

- codifica di brani musicali
- generazione automatica di strutture musicali
- elaborazione di brani musicali
- gestione della libreria di pezzi musicali
- didattica della musica
- analisi musicale

Una delle caratteristiche principali del sistema Teletau è quella di possedere un tipo di interfaccia-utente abbastanza semplificata, soprattutto da un punto di vista dell'utilizzatore non-vedente. Quest'ultimo, infatti, trova evidenti difficoltà nell'operare mediante finestre, icone e altri oggetti informatici di questo tipo; la nuova stazione di lavoro risulta quindi avere una interfaccia di tipo colloquiale, organizzata in comandi e sottocomandi (per il tipo di operazione o elaborazione), ognuno dei quali è dotato di opzioni specificabili sequenzialmente (per l'intervento su sezioni del brano sia orizzontali che verticali).

Malgrado questa semplicità nei confronti dell'interfaccia si è cercato invece di non penalizzare tutte le altre caratteristiche di intervento musicale che sono richieste ad un sistema "general purpose". Non tutte le possibiltà offerte dal Teletau possono trovare posto in questo sistema che, essendo basato su personal computer, non può avvalersi di quelle caratteristiche di memoria e velocità proprie delle macchine di tipo "mainframe".

Rispetto al sistema precedente quindi, il nuovo è stato progettato per essere impiegato in ambiente MS-DOS, sia per la sua eventuale portabilità che per la diffusione che tale tipo di calcolatori hanno anche tra operatori non-vedenti.

Vengono ereditati, per così dire, dal sistema Teletau, i principi generali di funzionamento e di accesso anche se, come detto, sussistono alcune limitazioni. In particolare, le operazioni di calcolo ed elaborazione dei brani avvengono passando sul disco fisso del PC e non più soltanto mediante la memoria centrale; come pure certe sottofunzioni non sono caricate inizialmente in memoria ma devono di volta in volta essere richiamate dai supporti secondari.

Ciò non toglie che si ha la possibilità di usare completamente lo standard di codifica Teletau per la codifica e la memorizzazione di brani musicali di qualsiasi tipo.

Il sistema di codifica consente all'utente di poter specificare l'altezza di un suono (notazione latina o anglosassone), la sua durata, la dinamica, il timbro e la sua disposizione nell'arco delle otto voci disponibili. Il codice è alfanumerico ma esistono comunque diverse possibilità alternative per la specifica sia delle frequenze che degli altri parametri acustico-musicali. Per esempio il LA centrale di durata un quarto può essere rappresentato come 4AQ

(codice simbolico) oppure come H 440 seguito dalla durata espressa in centesimi di secondo (codice numerico). Si possono anche inserire altezze non temperate e gruppi ritmici irregolari.

In ambiente MS-DOS risulta di una certa semplicità la gestione della libreria di pezzi musicali (comandi di Read e di Load) con la possibilità di essere eventualmente cambiati, elaborati, anche progressivamente e/o per sezioni, per mezzo di comandi appositi quali: Modify (consente totali o parziali modifiche di ogni paramero sonoro anche in maniera graduale, Goback (consente l'esecuzione al contrario, l'Invert (per l'inversione dei rapporti intervallari), lo Scale (per la specifica di un nuovo fattore di scala rispetto a quello caratterizzante le altezze temperate) e così via.

Non ultimi i comandi per la creazione di strutture musicali particolari (temperate e non) e la loro memorizzazione in un codice intermedio, tale da permettere l'eventuale evidenziazione di altezze, durate, dinamiche, timbri di uscita, etc.

Ogni fase dell'elaborazione musicale è caratterizzata dalla possibilità di creare file relativi: i file contenenti la codifica alfanumerica hanno l'estensione ".sou", quelli contenenti la codifica intermedia hanno l'estensione ".obj", quelli "suonabili" l'estensione ".mus".

Tutto quello che viene codificato o prodotto computazionalmente può essere suonato direttamente (comando Play) attraverso speciali routine software ispirate a quelle esistenti all'interno del programma musicale americano C.M.U. Midi Toolkit prodotto alla Carnegie Mellon University); queste possono "pilotare" una normale scheda MIDI del tipo Roland MPU-401 o compatibile ed un sistema di sintesi musicale MIDI di qualsiasi tipo e costo.

La stazione di lavoro basata su PC trova una sua ideale integrazione nel sistema DIFON2: si tratta di una scheda hardware per la lettura dei dati sullo schermo, con annesso un apposito sintetizzatore vocale; questa unità è stata progettata e realizzata da Paolo Graziani dell'Istituto per la Ricerca delle Onde lettromagnetiche del C.N.R. e già sperimentata in diversi ambienti lavorativi e scolastici che prevedono l'utilizzo del personal computer da parte del non-vedente. In pratica, ogni carattere spedito allo schermo viene automaticamente e istantaneamente inviato anche al sintetizzatore vocale: non solo tutto ciò che viene digitato sulla tastiera, ma anche eventualmente tutte le parole o simboli che appaiono sullo schermo possono essere quindi trasformati in indicazioni sonore per l'utente stesso.

BIBLIOGRAFIA

AA.VV., "TELETAU - A Computer Music Permanent Service", in **Proceedings of 1986 International Computer Music Conference**, (P. Berg ed.), San Francisco, California, Computer Music Association, 1986.

AA.VV., **Atti del I Convegno Nazionale Informatica, Didattica e Disabilità**, (a cura di C. Susini), Firenze, Italia, I.R.O.E./C.N.R., 1988.

AA.VV., **Computer Aided Production of Music Material for and by the Blind**, (P. Graziani ed.), Firenze, Italia, I.R.O.E./C.N.R., 1990.

Felician L., "Informatica e non vedenti: lo stato dell'arte", in **Rivista di Informatica**, vol. XVII, n.1, 1987.

TELETAU - Modalità Operative Rel 2.0, **Studi Musicali** (a cura di G. Nencini), Pisa, CNUCE/C.N.R., 1988.

ORION: A SINGLE CHIP DIGITAL SOUND PROCESSOR/SYNTHESIZER

***Giorgio Nottoli**

***Orla spa, Centro Ricerca e Sviluppo**
Via Dante Alighieri 22, 60022 Castelfidardo,Italy
Tel:+39-71-7808027 Fax:+39-71-976439

ABSTRACT

This paper describes a new sound synthesis/processing chip focusing on project goals and history, development methodology and applications.
The chip architecture is also described including key characteristics and performances extimation.
The Orion chip is completely microprogrammable and it's architecture is designed to optimize the execution of typical sound synthesis/processing algorithms. The basic primitives are realized by special hardware units designed for maximum execution speed.
Each main unit has the complexity of a subysystem and has an it's own local intelligence : more precisely it is able to execute elementary sequences of operations under the supervision of the main microcontroller.
Orion is constituted by four specialized arithmetic/logic units ,three data rams and three i/o units.
Executing music oriented digital signal processing algorithms the arithmetic work is distributed among the four alus to maximize computing parallelism while the data transfer work takes advantage of the three rams structure.
The whole project development was completed in july, 1990 when first samples was succesfully tested.
The Orion chip has one year of life and is actually employed in electronic musical instruments applications both as sound generator and audio processor.
Research and contemporary music applications are now in experimental stage with the idea to develop a fast, flexibile and cost effective real-time computer music system.
A single board two Orion system is to-day available for musical research applications.

Project goals and history.

The Orion project idea comes from the need to have a true music oriented digital signal processor chip as the basis of fast and cost effective computer music systems.
Contemporary music ideas need an high degree of flexibility and efficency : the composer needs to work very close to his own compositional principles, he wants to invent the total compositional and performance environment including sound generation.
Sound generation requirements incude tipically a large number of primitive elements as for example granular synthesis tecniques, thus computing efficency is a crucial parameter.
From one side, electronic musical instruments available on the market offer some sound generation efficency but poor flexibility, from the other side, commercially available digital signal processors offer the needed freedom but, in general have no sufficent efficency in computing music oriented DSP algorithms.
High efficents and flexible systems are available in some computer music center : the high complexity and costs limit the use to research and music production for a few people only but, the architecture of those discrete systems constitutes an important intellectual capital for to-day technology integrated circuits design, a reference point at the start of my work was certainly the 4X machine developed at IRCAM,Paris (Di Giugno,1980).
Those needs and considerations was fixed by the author mainly during the activity of SIM (Società per l'Informatica Musicale)srl, Rome.
The first 3 years (1982-84) was spent to develop various real time computer music systems around the Texas Instruments 320xx DSP family (Nottoli, Galante, 1986).
During the next 3 years (1985-87) the ASF (Audio Synthesis Family) chip set was designed and realized starting from an Elka spa - Texas Instruments Italy spa commitment (Nottoli, Galante,1988;Guarino, 1988).
During those 3 years I had the opportunity to learn extensively on how to make a chip and to experiment hightly pipelined architectures.
ASF is a very fast chip set, but it works as a fixed algorithms, only synthesis machine.
The Orion project was started from an Orla spa commitment with the goal to make a chip capable to efficently execute all the digital signal processing algorithms tipically used in an electronic musical instrument and to be a building block to make small low-cost systems and large high performance systems.
Those requirements was matching my needs about computer music : flexibility in algorithms definition and modularity for simple to complex systems design.
What very difficult to match was the high arithmetic precision needed for high quality computer music systems and the cost goal of an

industrial, cost effective chip : an hardly sufficent precision was chosen for the actual chip with the intention to enance it in the successive versions.
About six mounthes of work was spent to fix the architectural idea. This was done starting from a set of "target algorithms" and the related set of primitives. The final architecture was defined by means of a feed-bak design flow involving algorithms simulation on a software described processor and technology parameters such as gates speed, chip area, etc.
The result was a microprogrammable machine carachterized by a set of specialized computational units interconnected by an unidirectional multi-bus structure : something like a bridge betwheen special and general purpouse hardware.
Orion was developed during the last 2 years of SIM life (1988-89), completed in Sierra Semiconductors BV laboratories (s'Hetogenbosh, Holland) and applied to industrial products in Orla spa labs (Castelfidardo, Italy) starting from Summer 1990.

Development methodology and history.

The Orion chip was developed by means of two Apollo workstations as platforms for the Idea Mentor software package : an integrated environment to design ASICs (Application Specific Integrated Circuits).
A top-down design methodology was used partitioning the design in a set of blocks for each jerarchical level.
Heach functional block wad completed and simulated alone before introducing it in the whole context.
Special software was written to make and verify test vectors automatically : this approach saved a lot of design verification time and made better the system high-level menagement.
An Orion microassembler was also written both to support design simulation and chip applications.
A team of five people worked for one year in SIM srl, Roma until pre-layout simulations completed and, at the same time, a team of three people was working in Orla spa, Recanati to make applications microprograms and sound synthesis definition oriented to a complete line of electronic musical instruments.
Final simulations was done in SIERRA Semiconductors BV labs on a Vax system finalizing the design during six mounthes.
First Orion chip samples was received from the silicon foundery(SIERRA) at the end of july,1990 having 100% of functionality at the first time.
At that time the Orla spa Centro Ricerca e Sviluppo was constituted to support Orion chip applications and future versions development.

Key characteristics.

- 1.5 micron cmos process
- 68 pins PLCC package
- 32MHz clock, 62.5 ns microinstruction cycle.
- On chip microcontroller :
  - Writable 128x40 bits control store
  - 24 conditional jumps
  - One level of microsubroutine
  - 4 hardware loop counters
  - Always parallel data rams addressing and data move operations
- 3 Memories data storage architecture :
  - 2x128x20 bits data rams
  - 256x8 bits data/pointers/flags ram
- 4 Alus address/data processing architecture :
  - Pointers alu :
    - 2x pointer register
    - 2x increment register
    - Index register
  - Address/Ramp alu :
    - 36 bits adder/subtractor with result analysis flags
    - Double parallel comparator for ramp-stop computing
  - Signal processing alu :
    - Linear to exponential converter
    - 16x12 bits multiplier
    - 25 bits parallel shifter
    - 2 bits fractional shifter (1.5 dB step attenuator)
    - 20 bits saturation adder/subtractor and accumulator.
  - Table look-up alu :
    - Sine-wave generator
    - Fast linear interpolator
    - 24 bits address formatter/incrementer
- On chip input/output fuctions :
  - Dynamic ram controller with 16M words direct addressing
  - Bus compatible host microprocessor interface
  - 8M bits/sec. serial i/o interface
  - 4 bits microprogram controlled output port
- Multi-bus data move architecture : 104 bits /microcycle transfer rate

Key performances.

Performances of the Orion chip are a difficult figure to extimate becouse different applications permit different pipeline optimizations and need different amount of parameters data.
Actual experiences show that Orion performances are more limited by data storage capability then by execution speed.
Maximum speed optimization is permitted by highly iterative algorithms.
The examples shown below gives an idea of Orion performances when a simple algorithm is executed for the maximum possible number of iterations with a 40 KHz sampling frequency.
The number of possible iterations is shown as the performance extimation parameter. The needed processing to mix all the algorithms outputs and to send data to a stereo digital to analog converter is included in the examples.
A clock frequency of 32MHz is used to guarantee functionality for the entire commercial temperature and power supply voltage range.

Pcm synthesis : up to 31 pcm oscillators with independent current-point, end-point, loop-size, exponential envelope and level control.
The pcm samples are read with linear interpolation.

Fixed waveform synthesis : up to 42 oscillators with independent table look-up number, exponential envelope and level control.

Additive synthesis : up to 62 sine-wave oscillators with independent exponential envelope and level control.

Filters :

Second order two pole two zero IIR filter	: up to 31
Second order two pole IIR filter	: up to 62
Second order two pole one zero state variable IIR filter	: up to 18

Frequency modulation : up to 8 , 4 sine-wave oscillators Fm instruments with dynamic phase modulation interconnections between oscillators.

All the above synthesis methods and other too can be mixed together, in this case parts of different algorithms can be executed in parallel, for instance one pcm oscillator plus a second order two pole IIR filter takes only 8 microcycles (480 ns with a 32MHz clock) becouse the two algorithms can be executed totally in parallel.

Architecture overview.

Orion architecture is based on a 3 rams, 4 alus structore to balance arithmetic and data transfer timing in order to obtain maximum speed in music oriented digital signal processing.

The Orion chip is constituted by the following top-level units :

CLOCKS	: two phase clock generator
MICROCONTROLLER	:
X	: 128x20 bits ram
Y	: 128x20 bits ram
P	: 256x8 bits ram
P_ALU	: pointers processing alu
A_ALU	: address processing alu
S_ALU	: signal processing alu
T_ALU	: table processing alu
MPI	: host microprocessor interface
SIO	: serial input/output interface
EMI	: external dynemic memory interface

Orion is a microprogrammable machine, the included **microcontroller** is able to execute a wide set of microinstructions with high parallelism.
The chip activity is controlled by 3 different microinstruction fields: the first (named **jolly**) can control microprogram execution sequence, operation code for alus, input/output operations and control registers loading, the second (named **address**) controls internal rams address computing and the third (named **move**) controls data transfer between the Orion top-level units.
The microinstruction set includes **conditional jumps** also with autodecrement of hardware loop counters to execute iterative algorithms.
Microsubroutine call and return microinstructions are also included both with constant or variable address to allow dynamic algorithm allocation.
Characteristics of synthesizers or more generally sound signal processors that can be realized with the Orion chip are totally dependent from the microprogram and some external hardware, the sampling rate depends on number of microinstructions executed per sample : 60 ns each at full speed.
The Orion microprogram is stored in a ram named M (included in the microcontroller) the host microprocessor can download it halting the chip activity.
The M ram can contain up to 128, 40 bits wide microinstructions.

The 3 internal data rams X, Y and P store the parameters used by algorithms. Parameters can be related to internal rams addressing, internal sine-rom addressing, external memory addressing, signal description data, signal processing coefficents, execution control variables, etc.
The 3 rams are always accessible in parallel, simple to composed data types range from 8 to 48 bits size combining X,Y and P data sizes.
As an example a RAMP data type is 48 bits wide composed by the the 20 bits current ramp value, the 20 bits slope value and the 8 bits target value.

P_ALU computes the internal data rams addresses using pointers : adding a microprogram generated off-set to a pointer a "window" of data can be directly accessed from an algorithm.
A pointer can be updated loading it with an address value or autoincrementing it with the content of an increment register. Thus the data window can be moved anywhere on rams. Using autoincrement an iterative algorithm can operate on new data for each iteration.
P_ALU includes 2 pointers, 2 increment registers and a special pointer named Index Register : loading it from P ram dynamic interconnections between algotithms is possible.
Computing envelopes, the Index Register is loaded with a special control byte that is used to address sequentially an array of envelope segments data each constituted by target and slope values.
Using the above feature Orion can generate envelopes in a completely independent mode.

A_ALU is designed mainly to compute phase addresses and envelopes.
It includes a 36 bits adder/subtractor with result analysis flags and two parallel comparators : operands are loaded on two 36 bits registers and one 8 bits target value register.
Three main type of phase addresses can be computed :
1)internal sine-wave phase with 20 bits precision
2)external memory single cycle waveform with 20 bits precision and 16 bits table look-up number (up to 65536 waveforms can be addressed).
3)external memory full pcm phase address with 36 bits precision to describe typically current-address, end-address and loop-size.
Phase or frequency modulation can be computed adding a modulant signal value to the current phase or step value respectively.
To compute envelopes A_ALU compares current and final value to establish ramp direction (the slope value is always positive), then the ramp plus slope value is compared to target value to establish final result : the entire process takes 2 microcycles (120ns at 32MHz clock speed).
End of segment or end of envelope interrupts can be generated to request host computer service.

T_ALU executes four types of table look-up operations :
1)sine-wave generation using a built-in sine rom.
2)external memory 256 points table look-up.
3)external memory n points (up to 16M) table look-up.
Second and third type imply two memory accesses and linear interpolation between subsequent samples : T_ALU includes a fast interpolation hardware that operates in pipeline with memory accesses.

S_ALU is signal processing oriented and is usable mainly to envelope a signal in various wais, to attenuate signals with a defined dB ratio and to mix different signal together. The alu can be used also to compute various type of digital filters, second order IIR filters typically.
S_ALU is constituted by two parts : SA and SB.
SA includes an envelope converter (linear to exponential), a digital attenuator (100 dB, 1.5 dB steps), a 16x12 parallel multiplier and a 25 bits barrel shifter : multiplier and shifter are used indipendently or as a pseudo floating point multiplier to enlarge the dynamic range.
SB includes a 20 bits adder/subtractor with saturation and a 20 bits accumulator.

Orion has three input/output units plus a programmable 4 bits output port.

MPI is accessible by the host microprocessor that can request status informations or transfer operations.
Status informations are related to interrupt and interface conditions.
The host can read interrupt pending code to identify the interrupt source (tipically an envelope).
Transfer operations in the two directions can be executed : the host microprocessor can read the content of a data ram location or write data to it. It can be done virtually at any time becouse the chip transparency can be microprogrammed making idle the data rams every n microcycles.

SIO is an 8Mbits/sec. serial sincronous interface that can be used to transfer data to/from another Orion chip, to digital/analog converters or from analog/digital converters.

EMI is an external memory controller that includes a refresh counter, an address multiplexer and logic to generate all the signal needed to control dynamic rams.
EMI works in conjunction with T_ALU and receives from it addresses, operation codes and control signals.
Memory cycle timing is microprogrammmable : the cycle is divided in 3 parts, each of these is triggered under microprogram conterol.

A **MULTI_BUS** stucture is used to move data among Orion units. multiple data transfer allows to move data to/from a unit at the same time. Each input of each unit has a set of possible data sources. The MULTI_BUS structure can be changed dynamically during microprogram execution in a set of possible configurations to optimize data transfer for a given microprogram segment.

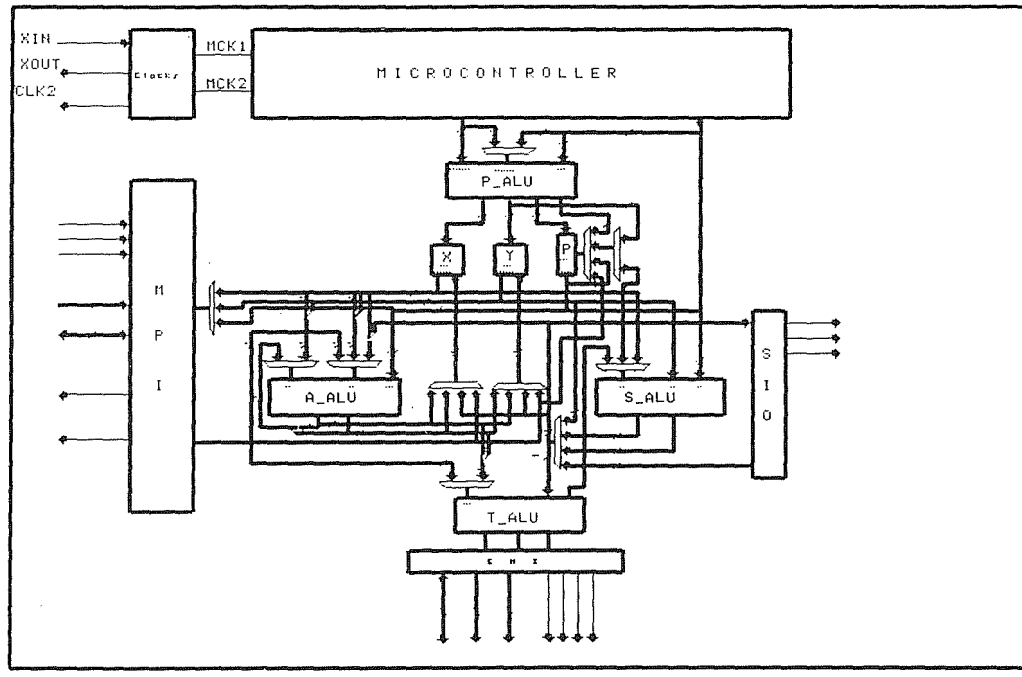


Figure 3-1 ORION's block diagram

To day and future industrial applications

The Orion chip based industrial products actually on the electronic musical instruments market are the most typical of Orla spa production: a complete line of electronic organs, electronic pianos and also an electronic "fisarmonica" (a very typical instrument of Castelfidardo music tradition).
The Orion chip was designed mainly for synthesizers, but it's flexibility permitted to solve the lot of different problems connected to those kind of instruments : special microprograms was written to simulate the futures of traditional electronic and alectro-meccanical organs, other was written to generate sounds with pcm, additive or frequency modulation tecniques and to process sound with reverberation and other algorithms.
Sound generation and processing systems designed for actual instruments use from two to six Orion chips each having a different microprogram.

At the present time about 20,000 Orion chips are runnig in those instruments, the major part in USA, England, France and Holland.

Future industrial applications will be electronic keyboards and professional synthesizers : those instruments are to day in the research and project definition stage.

Musical and research applications.

At the present time a relatively small research oriented system is available : it is constituted mainly by two cascaded Orion chips, 2M words of shared dynamic ram, two 18 bits digital to analog converters and one 16 bits analog to digital converter. All the hardware is contained in a PC-AT compatible board named **Alpha-Orionis**.
Basic software was developed to write, compile, load and run microprograms.
A set of c language macros and functions is also available to quick write musical and scientific application programs to control the Alpha-Orionis board in real time.
During the year 1990 two didactic and research programs was started : the first in the field of digital filtering tecniques at the "Università di Roma, Facoltà di Fisica", the second concerning real time analysis and resynthesis tecniques at "Università di Ancona Facoltà di Ingegneria".
Contemporary music applications are very difficult to plan and realize considering the to day's situation of contemporary music institutions and associations in our country.
I can only express an opinion : the lot of creativity and intelligence spent around technology must have a corrispondent artistic intelligence and creativity to realize what we call "computer music".

Future development.

Actually I am working on the second chip version.
The main inprovements will be an higher arithmetic precision for the signal processing part, more large internal memories and more speed. The architecture will be in general very similar to the actual one becouse results obtained until to-day by the Orion chip are an encouragement to run along the undertaken direction.

References

G.Nottoli, and F.Galante, "Soft machine: real time fully programmable computer music system", **Proc.ICMC**, Dhe Hague, Netherlands, CMA, 1986 pp.73-74.

G.Nottoli, and F.Galante, "A.S.F.:an audio synthesis family of VLSI chips", **Proc.VII Colloquio di Informatica Musicale**, Roma, Italy, AIMI, 1988 pp.37-41.

E.Guarino, "Tecnica di interpolazione multipla", **Proc.VII Colloquio di Informatica Musicale**, Roma, Italy, AIMI, 1988 pp.11-13.

V.Asta, A.Chaveau, G.Di Giugno, J.Kott, "Il sistema di sintesi digitale in tempo reale 4x", **Automazione e strumentazione, Volume XXVIII**, February 1980.

IL PROGETTO "LIVE": UN SINTETIZZATORE VIRTUALE E IL SUO UTILIZZO IN CONCERTO

Andrea Molino

CSC - Centro di Sonologia Computazionale
Università di Padova
via S.Francesco 11, 35100 Padova

ABSTRACT

With the developement of Computer Music technology, through which the commercial systems had become increasingly sophisticated and the software originally designed for mainframes had become more and more accessible, a musician can today find a large range of instruments of high quality and power. The difficulties that up until a few years ago composers came across, trying to realize on electronic instruments their musical ideas, decreased today so much that we can consider them to have virtually desappeared.

Due to this, the "ethical" contrasts which divided musicians (realtime/batch, commercial systems/mainframes etc.) appear today senseless; therefore the composer is obliged to assume a more conscious and "musical" position in regards to these kinds of instruments, excluding any "amateur" or "researcher" approach.

This paper describes a set of programs created for the realization of the computer part of a group of musical pieces belonging to the "Live" project. I illustrate the method I used to define the software and its operating characteristics, starting from the concept of "Virtual Synthesizer".

The software is actually composed by a Score Editor whose output code can be read by three groups of programs: the traditional Music5 system, an IBM-PC batch Sampler and a Standard Midi File Converter. Therefore the same input can be utilized by many various computer music actual instruments.

I finally describe an operating hypotesis about a concert performance system. Its most particular principle is the fact that it is HYBRID; the actual target is to balance the debits and the advantages of different kinds of instruments. The system is also modular, easily transportable and considerably cheap. Because of its "work in progress" nature, changes and improvements are allowed without being obliged to rebuild the system from the beginning.

1. INTRODUZIONE

Lo sviluppo delle tecnologie digitali applicate alla musica pone oggi il musicista di fronte ad una enorme quantità di sistemi di diverso tipo, con diverse caratteristiche e di diversa qualità. Se pochi anni fa la situazione era ancora di tipo "pionieristico", dove l'informatica musicale era terreno di sola ricerca scientifica più che di realizzazione artistica, oggi non solamente il mercato è invaso da strumenti musicali commerciali di basso prezzo e di qualità sempre più alta ma è possibile perfino disporre dei software più sofisticati e specialistici sullo schermo del PC di casa propria. Se personalità musicali di altissimo livello hanno sostanzialmente sacrificato, anni fa, la loro vita creativa a causa della necessità di sviluppare i mezzi tecnici adeguati alle loro esigenze musicali, oggi questa rinuncia non è più necessaria, tenendo anche conto del fatto che la formazione professionale dei musicisti include molto spesso la pratica viva con strumenti digitali, quando non addirittura una specifica preparazione parallela o conseguente alla preparazione "tradizionalmente" musicale.

Ciò apre la strada a metodologie nuove: si rende indispensabile la definizione di una *strategia* di lavoro in relazione alle necessità musicali di ciascun compositore. In un recente passato, ed in realtà anche oggi, si assisteva a scontri più o meno "ideologici" tra i fautori delle varie "filosofie" nel campo dell'informatica musicale; se voglio qui riprendere in mano la questione è perché questa comunicazione si basa sul presupposto che, proprio grazie a quello sviluppo tecnologico descritto in precedenza, prese di posizione di carattere "etico" non abbiano più ragione di esistere, per questioni essenzialmente *pratiche*. Ritengo inoltre assolutamente vitale che ogni ipotesi tecnica e soluzione operativa sia una *conseguenza* delle idee e necessità musicali; non è più ammissibile che un compositore rinunci alle proprie idee a causa dell'incapacità di realizzarle adeguatamente o, peggio, che la sua immaginazione sia limitata, talvolta in modo drastico, dalle caratteristiche e quindi dalle limitazioni di una particolare macchina o di un particolare sistema.

Questa comunicazione illustra allora le soluzioni che ho adottato per la realizzazione della parte elettronica e informatica dei brani musicali che compongono il progetto "Live", al quale sto lavorando in questo periodo. Il sistema che sto per descrivere è stato utilizzato per la realizzazione e l'esecuzione dal vivo di due di questi lavori: "Life Songs", per un percussionista e sistema digitale, programmato per il Festival '91 di "Antidogma Musica" a Torino, e "Five Live", per ensemble strumentale e sistema digitale, per la stagione di "Spazio Novecento" a Cremona.

Punto di partenza di questo lavoro è stata dunque l'esigenza di realizzare con la massima pienezze idee e necessità musicali che, per metodo, hanno *preceduto* ogni considerazione di carattere tecnico. Per sua natura, quindi, il sistema è *personale*; non tanto nel senso che non sia utilizzabile da altri musicisti, quanto nel senso che è costruito "su misura" sulle mie esigenze musicali. In realtà la speranza è che ogni musicista dovrebbe essere in grado di progettare e realizzare un sistema adattato alle proprie necessità.

Affronterò due campi di lavoro distinti e comunicanti: la *realizzazione* di musica con apparecchiature digitali e la sua *eseguibilità* dal vivo in concerto. Per quanto riguarda quest'ultima questione non esito ad affermare che ho violenta antipatia verso l'uso esclusivo del nastro magnetico come supporto per l'esecuzione dal vivo: la sua rigidità ha infatti posto enormi problemi di carattere musicale. Il mio obiettivo in questo senso è stato ed è dunque quello di poter gestire un'esecuzione dal vivo con la dovuta malleabilità, caratteristica questa indispensabile per rendere a pieno la *teatralità* alla quale rinunciare equivarrebbe per me ad un suicidio.

Il sistema descritto si compone allora di un pacchetto di software destinato tanto alla realizzazione quanto all'esecuzione efficace dal vivo di un lavoro musicale. La scelta di fondo ipotizza un sistema *integrato* e volutamente *misto*, impuro, sgombrando il campo da quelle prese di posizione di carattere "ideologico" alle quali ho già accennato; al contrario sono convinto che proprio la sua "impurità" potrebbe costituire una chiave per la sua efficacia.

2. "LIVE": UN SINTETIZZATORE VIRTUALE.

Per "sintetizzatore virtuale" intendo un sistema capace di leggere una partitura informatica organizzata secondo criteri arbitrari e di restituire un output costruito nei linguaggi comprensibili alle varie macchine che di volta in volta si ritenga opportuno utilizzare per l'esecuzione, con i compromessi strettamente necessari e senza che l'utente debba ogni volta risolvere i problemi *tecnici* pertinenti ad ogni sistema. È evidente come un sistema di questo tipo divenga indispensabile quando si adottino gli assunti di base di cui ho parlato in precedenza; il problema principale è infatti proprio quello di essere in grado di mantenersi, per così dire, al di sopra delle macchine utilizzate, e di gestire queste ultime con efficacia in rapporto alle idee di partenza.

Proprio per questo motivo, per poter progettare un sistema che rispondesse ai requisiti voluti, ho posto particolare cura nella definizione e nel chiarimento del metodo *musicale* del quale volevo servirmi, evitando accuratamente di legarmi, in questa fase, ad alcun particolare sistema già esistente: anche, tra l'altro, per evitare i rischi connessi al rapidissimo invecchiamento di ogni sistema informatico o i problemi di carattere strettamente pratico. La *formalizzazione* di questi spunti musicali di metodo ha poi portato ad una definizione funzionale e precisa delle caratteristiche che avrebbe dovuto avere il software.

Per quanto riguarda le tecniche di sintesi utilizzate, l'ibridazione di tecniche diverse (e la combinazione di teniche sintetiche con tecniche non sintetiche, come il campionamento) è considerata particolarmente feconda per la ricerca timbrica.

Il modello timbrico generale è composto in prima approssimazione di due categorie di strutture: il "corpo del suono" relativo alla frequenza fondamentale, nel senso che la struttura si sposta in frequenza con la frequenza di riferimento; il "colore del suono", invece, è strutturato a "zone di energia", nel senso che la distribuzione dell'energia nelle zone dell'udibile non cambia con il cambiare della fondamentale, secondo un trattamento di tipo formantico. Ogni timbro utilizzato generalmente il risultato di una combinazione di queste due categorie. Inoltre il trattamento dei parametri timbrici è generalmente gestito tramite algoritmi: ad esempio, i tempi di attacco e di decadimento di un suono sono correlati alla frequenza, e il sistema calcola i tempi appropriati a partire da un valore generale indicato nel "preset" timbrico. Una diversa suddivisione funzionale prevede infine la gestione di un suono in differenti "zone" timbriche: è evidente infatti che il trattamento timbrico dell'attacco, ad esempio, di un suono percussivo vada gestito in modo differenziato rispetto al sostenimento e al decadimento dello stesso suono.

Inoltre per ciascuna tecnica di sintesi utilizzata (principalmente additiva, FM, AM) sono stati sviluppati algoritmi per il trattamento intelligente dei parametri: per esempio, nella sintesi additiva un particolare sistema prevede una relazione tra l'ampiezza generale del suono e la sua ricchezza timbrica, e un ulteriore sistema garantisce una adeguata e indipendente mobilità nel tempo delle parziali; nel caso della FM i paramentri come l'indice di modulazione sono gestiti in modo articolato in relazione sia alla frequenza che all'ampiezza, per garantire omogeneità timbrica al variare della frequenza.

Il software, chiamato "Live 1.0", in realtà è un complesso di programmi differenti e interagenti, ciascuno con compiti diversi. Anche nella realizzazione del software ho seguito il principio della modularità: ognuno dei programmi che sto per descrivere è infatti un programma compiuto e funzionante, che può essere anche utilizzato disgiuntamente dagli altri. L'intero gruppo di programmi è stato realizzato per computer PC-IBM e compatibili.

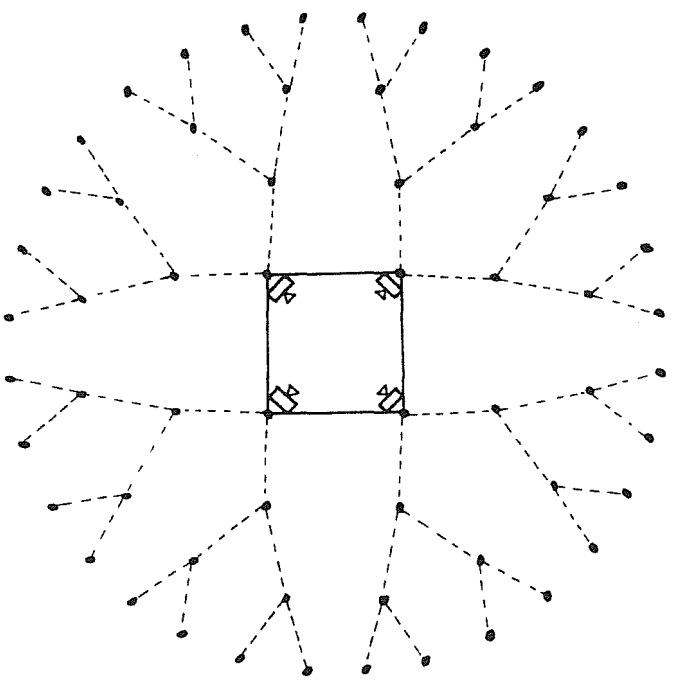
2.1. EdLive

Il compito principale del programma "EdLive" è la definizione della partitura, strutturata secondo criteri personali: ogni partitura (immaginata fino dal principio come una parte del lavoro completo, per consentire un ulteriore intervento in fase di mixaggio) è suddivisa in un numero arbitrario di cosiddette "figure", a loro volta suddivise in un numero arbitrario di "note"; alcuni parametri, come ad esempio il metronomo, sono applicati solo al livello più alto; altri, come ad esempio le caratteristiche timbriche degli

"strumenti", sono applicati solamente al livello delle "figure"; altri ancora attraversano invece l'intera struttura; la frequenza, ad esempio, si articola in una "frequenza madre", espressa in Hertz e definita al livello più alto, e in ulteriori valori definiti agli altri livelli, che si intendono come fattori moltiplicativi dei valori espressi ai livelli precedenti: modificando quindi un valore di questo tipo ad un certo livello si agisce in realtà anche su tutti i valori a livello più basso.

Ad ogni "figura" è anche possibile attribuire una coppia di valori di spazializzazione, secondo un arbitrario sistema che prevede quattro alberi binari che definiscono le "posizioni" nelle quali si può venire a trovare la "sorgente" attivata dalla figura in questione (fig1).

Fig. 1



Attraverso "EdLive" è quindi possibile definire singolarmente ciascun valore a ciascun livello; sono naturalmente possibili diverse opzioni di editing multiplo dei parametri. "EdLive" consente inoltre la definizione di grafici, (inviluppi, funzioni di controllo, spettri, ecc.) utilizzabili poi nella definizione della partitura, e di strutture di dati di vario tipo, come ad esempio preset timbrici, algoritmi e percorsi di spazializzazione, ecc., anch'essi utilizzati in seguito al momento della scrittura o della decodifica della partitura. Grafici e strutture dati vengono memorizzate in speciali directories accessibili poi alle altre funzioni del programma.

Al termine di questa elaborazione, "EdLive" restituisce in output un file di partitura, scritto in una directory SCORE e con estensione '.SCO, in un formato particolare che può poi essere convertito dal programma "TxtLive" in files di testo ASCII leggibili da tutti gli altri programmi che compongono il sistema.

Come è evidente, "EdLive" è il programma che più risente delle mie personali scelte e impostazioni musicali: in qualche modo, è un mio "ritratto" metodologico. Allo stesso tempo, ho cercato di evitare di trasmettere questa caratteristica ai programmi che avranno il compito di decodificare i files usciti da "EdLive"; a partire da questo punto, il compito del sistema è di *interfacciare* con la massima precisione la metodologia con la pratica.

2.2. TxtLive

"TxtLive" è, appunto, l'interfaccia tra Edlive e i vari sistemi implementati: produce files leggibili dal Music5, da SmpLive (campionatore in tempo differito su PC) oppure da MidiLive, che avrà in uscita files in formato Standard Midi File leggibili dai sequencers compatibili. La partitura viene letta, riordinata ed elaborata, e genera in output un file di testo ASCII destinato ad uno dei tre sistemi in una directory 'PAR' e con estensione, rispettivamente, '.PAR', '.SMP' oppure '.MID'. Per semplicità e per garantire una certa portabilità, il linguaggio ASCII usato per SmpLive e MidiLive è molto simile al tradizionale linguaggio Music5.

Da notare che questo stadio di elaborazione della partitura comprende anche l'esecuzione di una serie di operazioni come ad esempio la lieve deviazione random delle frequenze e dei tempi di attacco delle note; tali operazioni, essenziali per la qualità musicale del risultato, sono quindi risolte "per default", senza che il compositore se ne occupi *coscientemente*. Allo stesso modo, l'enorme quantità di dati necessaria per una gestione soddisfacente di software a basso livello, come ad esempio il Music5, viene risolta a livello di interfaccia, senza interessare l'azione diretta dell'utente; eliminando così il problema principale nell'utilizzo di sistemi di questo tipo.

2.3. SmpLive

"SmpLive" è un vero e proprio campionatore in tempo differito, su IBM-PC; naturalmente necessita di opportune schede di conversione DAC. Dopo aver letto e decodificato il file di testo proveniente da "TxtLive", "SmpLive" cerca in una directory SAMPLE, a seconda dello "strumento" indicato nella partitura in input, i relativi files di campioni pre-memorizzati e li utilizza come preset, trasponendoli e modificandoli in base alla partitura stessa. Un primo output è costituito da un file quadrifonico (diviso in realtà in due files stereofonici) contenente il segnale diretto e di un ulteriore file monofonico contenente il segnale da riverberare per la spazializzazione secondo un sistema derivato da Chowning che descriverò in seguito. Tale file può poi venire riverberato attraverso delle procedure apposite e sommato semplicemente ai due files precedenti. Un'altra ipotesi, più efficace dal punto di vista della spazializzazione ma più impegnativa per l'esecuzione in concerto è la riverberazione e mixaggio dal vivo del file del segnale da riverberare. La durata del preset è potenzialmente infinita, e NON dipende dalla capacità dello heap.

Una ulteriore versione di "SmpLive", chiamata "ISmpLive", implementa una procedura di reset della durata che evita la modifica della durata del campione in relazione alla trasposizione in frequenza.

Nell'area operativa di "SmpLive" si trovano anche dei programmi di riverberazione su PC, che ricevono in ingresso files di campioni con il segnale da riverberare e restituiscono ulteriori files di campioni con il segnale riverberato; gli effetti implementati sono l'eco semplice, l'eco ricorsivo (Comb), l'eco ricorsivo con filtro passabasso nell'anello di retroazione e l'allpass; i parametri dei riverberi vengono definiti interattivamente. Questa dotazione di base permette un numero virtualmente infinito di configurazioni; le più comuni di esse (4 Comb in parallelo e 2 Allpass in serie, ecc.) sono state implementate separatamente. Infine, due piccoli programmi (SV_DSP e DSP_SV) consentono la compatibilità di files audio normali in files audio secondo il formato usato dal software commerciale SAMPLEVISION, e viceversa, in modo da garantire la portabilità dei files tra il campionatore su PC e i campionatori MIDI disponibili in commercio.

2.4. MidiLive

"MidiLive", dopo la lettura ed elaborazione della partitura, genera un file in formato Standard Midi File destinato a normali sequencers compatibili con tale formato.

Un file ASCII di configurazione fornisce a "Midilive" informazioni sul sistema MIDI a disposizione, in modo da poter gestire secondo la partitura anche il parametro timbrico, attraverso i messaggi di Program Change, Control Change e System Exclusive. MidiLive è in grado di scrivere files Standard Midi di formato 0 e 1.

Vorrei sottolineare la possibilità, per l'utente, di bypassare l'utilizzo di "EdLive" e di scrivere, direttamente o attraverso altri eventuali programmi, i files di testo destinati al campionatore o al MIDI. Inoltre è espressamente previsto l'utilizzo del sistema in *simbiosi* con altri software presenti sul mercato, a partire dallo stesso Music5 per arrivare a Sequencers, Sound files editors, eccetera:

Il complesso di programmi "Live" è realizzato in Turbo Pascal versione 5.5. La struttura a librerie consentita dal linguaggio Pascal è particolarmente adatta alla concezione modulare del software: ogni singolo programma dispone infatti di un certo numero di librerie riservate e pertinenti al sistema specifico; ad esempio la libreria "M5Ins", utilizzata dalla sezione Music5 di "TxtLive", consente la gestione modulare e la scrittura su file degli strumenti Music5. Un altro blocco di librerie si situa invece a livello più alto, e mette a disposizione di tutte le parti più specifiche dei vari programmi una serie di procedure e funzioni di carattere generale: ad esempio la libreria "MuLib" consente operazioni di carattere musicale, come la conversione di valori di ampiezza dalla scala lineare alla scala in dB, servendosi a sua volta di procedure e funzioni ancora più generali, come le funzioni logaritmiche, contenute nella libreria "Func", che si situa in cima alla struttura.

3. UN SISTEMA MISTO PER L'ESECUZIONE IN CONCERTO

A questo punto, diviene vitale la definizione a priori di un sistema digitale da concerto sufficientemente sofisticato da poter affrontare con la dovuta efficacia e malleabilità le necessità connesse con l'esecuzione, tenendo conto del fatto che la partitura è stata stesa, per principio, senza tenere conto delle limitazioni dei sistemi tradizionali (nastri magnetici, ecc.), e che la presenza di strumentisti dal vivo e la loro continua interazione dinamica con il sistema digitale è una caratteristica spettacolare ed emotiva alla quale non intendo rinunciare per nessun motivo. D'altro canto, è necessario che il sistema non sia eccessivamente impegnativo dal punto di vista organizzativo, per permettere un trasporto agevole e con costi il più possibile contenuti.

La scelta di base, per così dire strategica, è stata anche qui di non immaginare in partenza un sistema eccessivamente rigido, e di prevedere fino dal principio di poter risolvere alcuni specifici problemi pratici a seconda delle apparecchiature di volta in volta disponibili, senza per questo dover modificare per questo la partitura. Si tratta dunque, fatte le dovute distinzioni, di una sorta di "orchestrazione" operativa, migliorabile e modificabile a seconda delle situazioni.

In base a queste premesse, l'ipotesi operativa che segue è flessibile e generale; quindi suscettibile di aggiustamenti, anche in relazione all'evoluzione della tecnologia e alla maturazione di nuove necessità musicali. Infine, l'ipotesi qui definita non esclude a priori alcun tipo di sistema: la strategia scelta al contrario quella di compensare i limiti di ciascun tipo di sistema con i vantaggi degli altri, nella convinzione che dualismi fino ad ora oggetto di polemiche per così dire "etiche", come ad esempio tempo reale/tempo differito, sistemi personali/grandi sistemi, sintesi/campionamento, e via discorrendo, siano del tutto artificiosi e comunque pertinenti alla sfera *tecnica* piuttosto che a quella musicale, che invece l'obiettivo primario del mio lavoro.

Un punto di partenza metodologico è la considerazione che sistemi particolarmente agili e duttili, come i sistemi basati sul codice MIDI, si trovano in grande difficoltà al momento di gestire una quantità

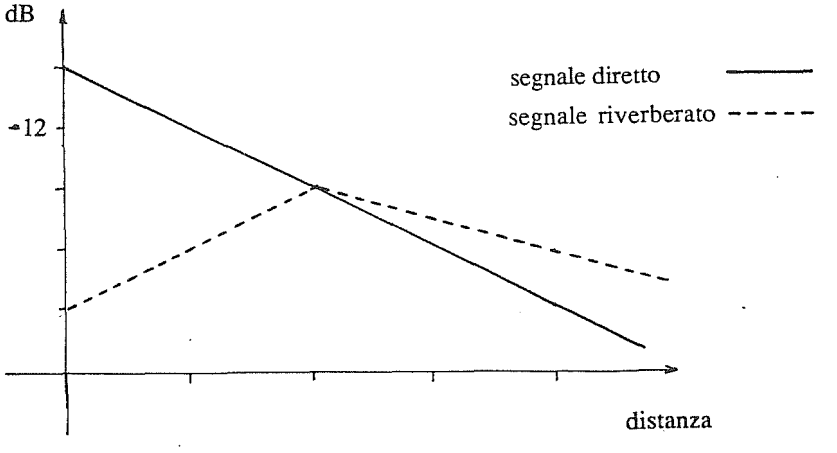
molto alta di dati in tempi molto brevi; d'altro canto, i sistemi in grado di elaborare efficacemente un numero arbitrario di dati in tempi brevi operano per lo più in tempo differito, e obbligano perciò ad una gestione dal vivo più rigida e meno controllabile in concerto. Un soluzione efficace appare allora un sistema *misto* risultante dall'*interazione* di sistemi dei due tipi; la partitura potrà allora venire realizzata suddividendo le varie parti, a seconda delle loro caratteristiche, tra le varie parti del sistema.

Il sistema utilizzato per le esecuzioni già citate al principio di questa comunicazione era composto quindi da due distinte workstations, basate rispettivamente su un sistema DSP batch su PC dotati di schede di conversione DAC e su un complesso di apparecchiature MIDI, di carattere commerciale; il tutto è integrato, oltre che dal software già descritto, da un'ampia scelta di software commerciali. L'azione combinata delle due workstations permette così una realizzazione efficace della partitura. In linea di massima, ad esempio, gli eventi che prevedono un numero elevato di sincronizzazioni temporali con gli esecutori dal vivo vengono eseguiti principalmente su strumenti in tempo reale che implementano il sistema MIDI; la gestione delle situazioni più complesse e nel contempo meno impegnative dal punto di vista della sincronizzazione è invece affidata affidata al primo sistema.

Naturalmente, con la diminuzione dei costi ed il progresso tecnologico il sistema sarà soggetto a miglioramenti, che sono stati previsti fino dal principio; la modularità del sistema è stata volutamente accentuata proprio per garantire questa possibilità. L'insieme di queste apparecchiature, unito ad un normale sistema quadrifonico di mixing e diffusione sonora e ai microfoni per la presa in diretta del suono degli strumenti dal vivo, ed eventualmente ad un registratore multitraccia, costituisce la dotazione prevista per un concerto. Come si può vedere, il complesso di apparecchiature è facilmente trasportabile e ragionevolmente agile; questo garantisce immediatamente una caduta notevole dei costi a carico delle associazioni di concerti che intendano servirsene, e permette così una maggiore diffusione dei lavori musicali.

Le necessità di spazializzazione già descritte sono state affrontate attraverso un approfondimento dell'algoritmo elaborato da John Chowning. Una serie di test ha permesso di individuare un algoritmo di variazione dei parametri che appare particolarmente efficace. Ad ogni step di "distanza" il segnale diretto viene attenuato di 12 dB, *senza per questo modificarne le caratteristiche timbriche*; la quantità del segnale originale da inviare ai riverberatori è calcolata secondo il grafico di Fig. 2.

Fig. 2



Per quanto riguarda l'applicazione di questo algoritmo agli strumenti MIDI è possibile suddividere il segnale in due distinti canali del mixer digitale, e di gestire i livelli di segnale direttamente via MIDI tramite sequencer.

4. CONCLUSIONI

Come ho già ampiamente detto in precedenza, questo contributo vuole essere l'illustrazione di una esperienza personale, rispondente a necessità personali, piuttosto che un'ipotesi operativa generica, valida in astratto. Tuttavia posso immaginare che le esigenze musicali e organizzative che mi hanno spinto a riflettere in modo radicale sul problema e a tentare di definire una strada per la sua soluzione siano comuni a molti dei musicisti che decidano di servirsi di strumenti musicali digitali. Il realtà il mio auspicio è che la preparazione tecnica e informatica media dei musicisti divenga tale da consentire loro di progettare e costruirsi un sistema, per così dire, "su misura", esattamente come quando un compositore sceglie un organico strumentale piuttosto che un altro in relazione alle proprie idee e necessità musicali. Tutto ciò parte dall'assunto di principio che il "tempo dei pionieri" è ormai terminato; che, per chi voglia fare musica, è ora di pensare alla musica.

5. BIBLIOGRAFIA

J.Chowning: "The synthesis of complex audio spectra by means of frequency modulation", Journal of the Audio Engineering Society, 1973

J.Chowning: "The Simulation of Moving Sound Sources", ibid., 1971

G.De Poli: "Tecniche numeriche di sintesi della musica", bollettino LIMB n. 1, Venezia, 1981

G.De Poli-A.Vidolin: "Manuale Music5", rapporto interno CSC, Padova, 1988

J.M.Grey: "An Exploration of Musical Timbre", Tesi, Stanford 1975

M.Mathews: "The Technology of Computer Music", The MIT Press, Cambridge, USA, 1969

J.A.Moorer: "About this Reverberation Business", Rapport IRCAM, Parigi 1978

J.C.Risset: "An Introductory Catalog of Computer-Synthesized Sounds", Bell Laboratories, New Jersey, USA, 1969

A.Vidolin: "Interazione fra i livelli di rappresentazione dell'informazione musicale nella composizione mediante computer", Automazione e Strumentazione, 1980

Capitolo 5: Elaborazione Numerica di Segnali

A NEW CONTRIBUTION TO NOISE ANALYSIS OF THE INTERPOLATING SINUSOIDAL DIGITAL OSCILLATOR

Francesco SCALCON

C.S.C. - Centro di Sonologia Computazionale
D.E.I. - Dipartimento di Elettronica e Informatica
University of Padua, Via Gradenigo 6/a, 35131 PADUA, Italy

Keywords: Table Lookup Oscillator, Sinusoidal Signal Generation, Interpolating Noise, Phase-jitter.

ABSTRACT

In this work we have analysed the quality of a sinusoidal signal, generated by a digital oscillator, calculating the SNR (Signal-Noise Ratio). A technique is proposed to obtain the same SNR of a rounding oscillator, using the hardware of a truncating oscillator. The influence of the interpolation bit number on the SNR for an interpolating oscillator has been also studied. Moreover, we have studied a complete model of the digital oscillator that takes into consideration all the cases regarding the following operations of sampling, quantization and interpolation. The relationships developed regarding this model show that the best distribution of the noise among the spectral lines can be obtained if the value of the increment register is odd.

INTRODUCTION

There are different methods to obtain a sinusoidal digital oscillator. The most commonly used is the table lookup method, in which a suitable number of equispaced samples of a sine function period are written in a table. While doing the synthesis the samples are read with an appropriate sampling increment in order to achieve the desired frequency (Mattews, 1969). The evaluation of the quality of the signal produced by an oscillator can be supplied by the SNR and the spectral analysis. With regard to the SNR, it is defined as follows:

$$SNR = 10 \log_{10} \frac{\sum_{0}^{N-1}{}' {}_k \; s_k^2}{\sum_{0}^{N-1} {}_k \; (s_k^* - s_k)^2} \qquad (1)$$

where s_k^* is the digital signal generated by the oscillator and s_k is the ideal digital sinusoidal signal:

$$s_k = \sin (2\pi f k / F_c) \qquad (2)$$

An article about the SNR has already been published (Moore, 1977). With our work we propose some new results.

SNR REDUCTION

In this work a technique to improve the SNR of the truncating oscillator is proposed. Normally, the table of this digital oscillator contains the samples:

$$T_i = \sin(2\pi i/N) \qquad i = 0,\ldots,N-1 \qquad (3)$$

so, in each interval between a sample and the next, the sine function is approximated by its value in the left extreme (see fig.1a). Instead of the samples (3), we suggest to fill in the table with the values l_k^*, which are the solutions of the problems:

$$\min_{l_k} \int_{k\frac{2\pi}{N}}^{(k+1)\frac{2\pi}{N}} (\sin x - l_k)^2 \, dx \qquad (4)$$

These solutions are:

$$l_k^* = \frac{N}{2\pi}\left[\cos(2\pi k/N) - \cos(2\pi(k+1)/N)\right] \qquad (5)$$

or in other terms

$$l_k^* = \frac{\sin(\pi/N)}{\pi/N} \sin(2\pi k/N + \pi/N) \qquad (6)$$

These values minimize the least mean square error. In this way to every phase value of the interval

$$[2\pi k/N,\ 2\pi(k+1)/N[$$

corresponds the value l_k^* (see fig.1b): the least mean square error reduces and the SNR increases. By (6) we can see that l_k^* is near the sine function valuated in the midpoint of the interval (with N great enough the multiplying factor

$$\frac{\sin(\pi/N)}{\pi/N}$$

is close to 1).

It is easy to realize that the LSM error is very similar to the one which would be obtained by the rounding oscillator. We have verified experimentally that both a truncating oscillator using the samples (6) and a rounding oscillator reading a commonly used table (3) produce a signal with the same SNR.

INTERPOLATING BIT NUMBER

As far as the interpolating oscillator is concerned, we can wonder: is it necessary to utilize all the bits of the fractionary part of the phase register for the linear interpolation? Or is it sufficient to utilize only some of the most significant? In which way does the interpolation bit number vary the SNR?

Some suitable algorithms have been written to generate the sequence

s_k^* by a general-purpose computer, while the ideal sequence s_k has been computed using the highest accuracy (80 bits floating point variables) of the same general-purpose computer.
The algorithm for the calculation of SNR of interpolating oscillator supplies the SNR by using three parameters: memory length, memory word-length and interpolation bit number. From the results of this algorithm we can observe that, once fixed the table dimensions (memory length and memory word-length) there is an "optimum" number of interpolation bits: these are reported in Table 1. The meaning of this "optimum number" is the following. Taking as an example a table of 1024 words of 8 bits, if we use 3 bits for the linear interpolation, we obtain a SNR of 58 dB. Even if we use more than 3 bits, the SNR does not increase sensibly.

INTERPOLATING OSCILLATOR MODEL
In 1983, S. Mehrgard derived the spectra of sinusoidal signals generated by a rounding oscillator. Here we attempt to extend the same analysis to an interpolating sinusoidal oscillator.
The model of an interpolating sinusoidal digital oscillator is shown in fig.2.
Let F be the sampling frequency,
N be the table's length,
B be the bit number of samples,
K be the interpolation bit number,
f_o be the digital oscillator frequency,
I be the value of the increment register.
We obtain f_o by:

$$f_o = F_c I/N \tag{7}$$

Let the input signal be a sinusoidal function:

$$s_A(t) = \sin 2\pi f_o t \tag{8}$$

The output signal will be equal to the one generated by an interpolating digital oscillator.
Oscillator parameters and frequencies of the system are related by:

$$F_2 = F_c \tag{9}$$

$$F = F_c I \tag{10}$$

$$F_1 = 2^K F_c I \tag{11}$$

$$f_o = F_c I/N \tag{12}$$

To simplify, we disregard the effect of quantization.
In A the frequency representation is given by the well known formula:

$$S_A(f) = \frac{1}{2j}\left[\delta(f-f_o)+\delta(f+f_o)\right] \tag{13}$$

In B the sinusoidal signal has been sampled; because of this sampling

its frequency representation is given by:

$$S_B(f) = \frac{1}{2j} \sum_{-\infty}^{+\infty}{}_n \delta(f-(nF-f_0)) + \delta(f-(nF+f_0)) \qquad (14)$$

In C the sampled signal has been reconstructed by a first order holder, whose frequency response is given by:

$$H_1(f) = \frac{1}{F} \cdot \left[\frac{\sin \pi f/F}{\pi f/F} \right]^2 \qquad (15)$$

Spectrum becomes:

$$S_C(f) = \frac{1}{F} \frac{1}{2j} \sum_{-\infty}^{+\infty}{}_n A_n \delta(f-(nF-f_0)) + B_n \delta(f-(nF+f_0)) \qquad (16)$$

where $A_n = \left[\frac{\sin \pi(nF-f_0)/F}{\pi(nF-f_0)/F} \right]^2$ and $B_n = \left[\frac{\sin \pi(nF+f_0)/F}{\pi(nF+f_0)/F} \right]^2$

Spectrum is shown in fig.3a.
Now, if we sample the signal at the frequency F_1, spectrum becomes periodic again at the frequency $F_1 = 2^k F$. This sampling does not create new bars; they are the very same we can see in the frequency representation of fig.3a. Since F_1 is multiple of F, every translation of fig.3a will have lines with frequency given by

$$nF \pm f_0 \qquad (17)$$

as we can see in fig.3b (where k=1).
In D, the signal spectrum is still characterized by bars, whose frequencies are given by (17).
In E, the signal has been reconstructed by a zero order holder; the holder transfer function is:

$$H_0(f) = \frac{1}{F_1} \frac{\sin \pi f/F_1}{\pi f/F_1} \exp(-j\pi f/F_1) \qquad (18)$$

Now, the spectrum is represented as follows:

$$S_E(f) = \sum_{-\infty}^{+\infty}{}_n A'_n \delta(f-(nF-f_0)) + B'_n \delta(f-(nF+f_0)) \qquad (19)$$

Finally, the signal is sampled again at the frequency F_2.
Since the signal becomes periodic again, in G the frequencies of spectral lines may be written as

$$(nF \pm f_0) + mF_2 \qquad (20)$$

where n and m are integer.

SPECTRAL LINES SPREADING

Let's now take a look to the spectrum and consider the range between 0 and F_2. The number of bars belonging to this range is finite because the signal generated by the oscillator is periodic. Now, we can prove that the lines are in number of r, where

$$r = 2^{\max(a,b)-a+1} \qquad (21)$$

where:

a is the molteplicity of 2 in the decomposition of the value of increment register, regarded as integer;

b is the bit number of the fractionary part of increment register.

We can therefore easily prove that if the increment register value is odd (a=0), then the number of bars is maximum: the energy concerned with the noise shares itself through the highest number of bars.
From a psycoacoustical point of view, under the same SNR the number of discrete noise frequencies should be maximized in order to obtain the best results in terms of perceived noise. In other words, we should avoid to concentrate the noise on one or few frequencies (this practice of maximizing the number of noise bars is commonly used in the field of telecommunications). So it is useful to try and put "1" into the less significant bit of increment register but keeping the requested frequency precision.
The length of increment and phase register are decided using the "worst case" method. We must pay attention to the precision of the last significant bit in some "critical cases" only (few sampled tables, low frequency). Elsewhere the previous bit may assure the required precision. Since we are rarely concerned with critical cases using a sinusoidal digital oscillator, we can easily put "1" into the less significant bit of the increment register.

REFERENCES

M.V. Mattews, The Technology of Computer Music, Cambridge MA, M.I.T. Press, 1969.

F.R. Moore, "Table Lookup Noise for Sinusoidal Digital Oscillator", Computer Music Journal, vol.1, n.2, April 1977, pp.26-29.

S. Mehrgardt, "Noise Spectra of Digital Sine-Generators Using the Table-Lookup Method", IEEE Trans. Acoust., Speech, Signal Processing, vol.ASSP-31, n.4, August 1983, pp.1037-1039.

Memory lenght	Memory wordlength in bits (not including the sign bit)										
	8	9	10	11	12	13	14	15	16	17	18
32	6	6	6	6	6	6	6	6	6	6	6
64	7	8	8	8	8	8	8	8	8	8	8
128	6	7	8	8	9	9	9	9	9	9	9
256	5	6	7	8	9	9	10	10	10	10	10
512	4	5	6	7	8	9	10	10	10	11	11
1024	3	4	5	6	7	8	9	10	11	12	12
2048	2	3	4	5	6	7	8	9	10	11	12
4096	2	2	3	4	5	6	7	8	9	10	11

TABLE 1: "optimum number" of interpolation bits. A higher number of interpolation bits does not improve sensibly the Signal-Noise Ratio.

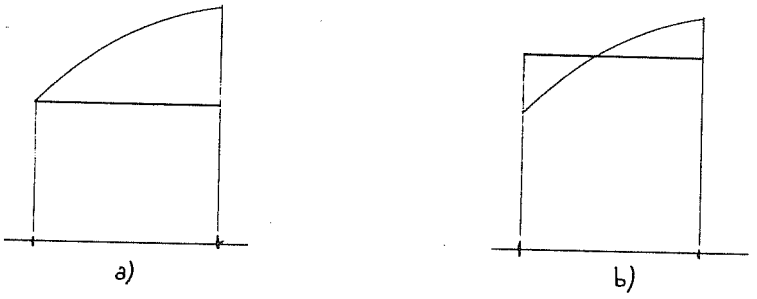


fig.1: a) approximation of a sine function interval with a commonly used table
b) approximation of a sine function interval with an optimized table

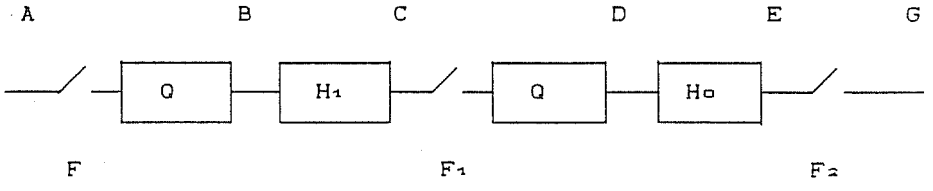


fig.2: block diagram of an interpolating digital oscillator model

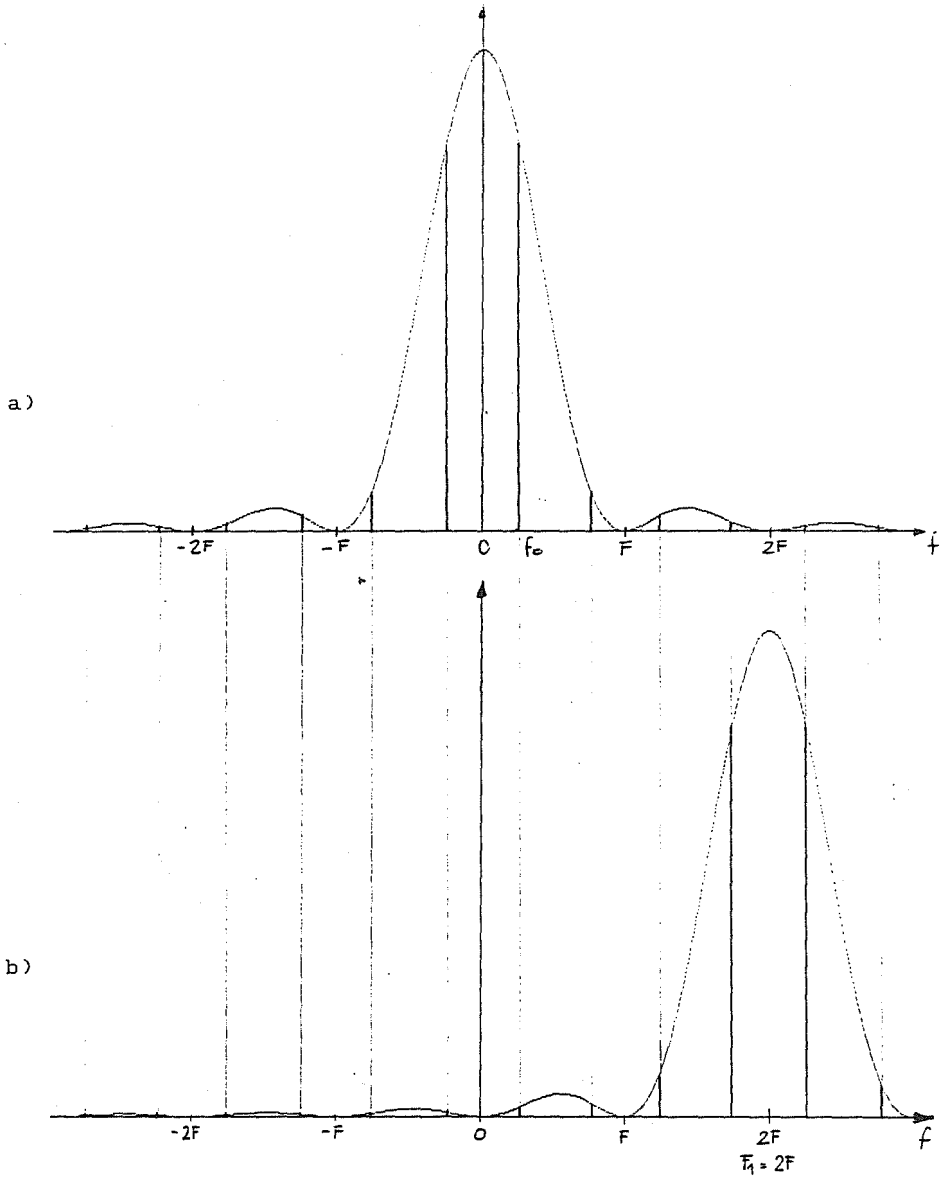


fig.3: a) spectrum of signal in C
b) translation of the previous graph (with $F_1 = 2F$)

UNA CLASSE DI ALGORITMI PER ECCITATORI DINAMICI NON LINEARI

Giampaolo Borin, Giovanni De Poli, Augusto Sarti

Centro di Sonologia Computazionale
Dipartimento di Elettronica e Informatica
Via Gradenigo 6a, 35131 Padova
Email: depoli@elett1.unipd.it
telefono: 049/8287631, fax: 049/8287699

Abstract

The interaction between excitator and resonator is a well established concept which critically deternimes the timbral evolution of the sound produced by a natural instrument. We believe that a scheme based on this interaction can be usefully applied to digital synthesis techniques, in order to obtain the timbral dynamics as a structural property of the algorithm, rather than the result of very complex actions on the parameters. For this reason we proposed and analyzed in a previuos work a synthesi scheme based on the interaction of two dynamic systems representing a generalized excitator and resonator.

In this article the attention is focused on the excitator, which represents the most critical part of the algoritm, being strongly non linear and in general with memory. We propose a structural scheme for the excitator characterized by certain characteristics of regularityand low computational cost. The main purpose of this scheme is the functional modularity, which allows us to define the structure of the excitator with specific reference to the resulting algoritm. To show the degree of generality of the proposed excitator structure we present and analyze several examples of applications belonging to the class of physical, pseudophysical and non-physical algoritms.

1. Introduzione

Ogni strumento musicale puo' essere utilmente descritto facendo riferimento ad un semplice schema basato sulla interazione di un eccitatore e un risuonatore. Per operare la separazione tra eccitatore e risuonatore si puo' far riferimento alle loro caratteristiche funzionali, intendendo per eccitatore la parte che causa e in certi casi sostiene il fenomeno vibratorio nel risuonatore e per risuonatore la parte dove hanno luogo i fenomeni vibratori che rendono il suono musicalmente interessante. L'interazione tra eccitatore e risuonatore e' un fattore determinante per la ricchezza dinamica dell'evoluzione timbrica del suono prodotto.

Noi riteniamo che lo schema che prevede l'interazione tra eccitatore e risuonatore, oltre ad essere un utile modello per gli strumenti musicali acustici, puo' essere interpretato in una maniera piu' generale. La separazione tra eccitatori e risonatori puo' infatti essere utilmente estesa anche alle tecniche numeriche di sintesi. In tale modo la dinamica timbrica del suono puo' essere ottenuta come una proprieta' strutturale dell'algoritmo piuttosto che come un risultato

delle azioni di controllo nell'esecuzione o delle variazioni dei parametri.

In (Borin et al. 1989,1990a) abbiamo proposto uno schema per la simulazione degli strumenti musicali basato tre blocchi E, I, R: un blocco E per l' eccitatore, costituito da un sistema dinamico, causale, generalmente non lineare e tempo variante; un blocco I di interconessione; un blocco R per il risuonatore, costituito da un sistema dinamico, causale, generalmente lineare e tempo variante.

In questo lavoro prendiamo in considerazione la struttura dei vari blocchi. In particolare si puo' osservare che la struttura dell' eccitatore, essendo non lineare, e' piu' difficile da analizzare e implementare. Fra tutte le possibili strutture bisogna cercare uno schema strutturale che sia adatto per un gran numero di possibili eccitatori e che sia utile per una implementazione hardware. Cio' significa che, oltre ad un certo grado di generalita', la classe di algoritmi risultanti deve essere caratterizzata da un basso costo computazionale e da opportune caratteristiche di regolarita' strutturale.

Il problema di caratterizzare la struttura dell'eccitatore , dal punto di vista algoritmo, é stato affrontato anche da Florens (1990). Egli inserisce questo problema nell'approccio modulare sviluppato dall'ACROE (Florens&Cadoz, 1991), distinguendo cioé elementi di massa puntuali ed elementi di collegamento e concentrando la non linearità in questi ultimi. Gli elementi massa ricevono in ingresso informazioni di forza e ritornano posizione, quelli di collegamento invece ricevono informazioni di posizione e ritornano forza. Nelle connessioni vengono implicitamente distribuiti i ritardi per consentire la computabilità. Con questo approccio Florens ha descritto i vari tipi di eccitazioni percussive, pizzicate e sostenute. D'altra parte questo concetto di modularità pone anche vincoli sul tipo di struttura che può essere descritto. Un' altro approccio di tipo modulare e' la sintesi modale (Adrien 1991). Lo strumento fisico viene diviso in varie parti interagenti. Ogni parte, supposta lineare, viene modellata con il formalismo modale attraverso la realzione ingresso uscita di coppie di grandezze estensivo/intensivo e si pongono le non linearita' nelle connessioni. Resta poi da risolvere sistemi non lineari per trovare le variabili mutuamente dipendenti.

Noi invece consideriamo la modularità da un punto di vista funzionale, più che dalla struttura fisica. Funzionalmente distinguiamo una parte eccitatore da una parte risuonatore e descriviamo ciascuna parte nel modo più appropriato. Questo ci consente di definire la nostra struttura direttamente con riferimento all'algoritmo. In questo modo si evita che i vincoli sulla struttura fisica e quelli che derivano nel passaggio da struttura fisica ad algoritmo, diminuiscano la potenzialità di quello che é possibile ottenere. Si ha così un approccio molto più generale che non solo consente la simulazione dei più tipici eccitatori usati negli strumenti musicali, ma può essere usato per creare altri algoritmi. L'obiettivo é cercare di individuare una struttura di algoritmi che condividano le motivazioni profonde del funzionamento degli strumenti fisici. In questo modo si può sfruttare, almeno in parte, l'esperienza di secoli di liuteria come fonte di ispirazione e guida nel pressochè inesplorato campo degli algoritmi non lineari con reazione per la sintesi dei suoni.

2. Una classe di algoritmi per l'eccitatore.

Va rilevato che esiste una importante differenza dello schema feedback o di interazione quando si passa dalla realta' fisica (continua) alla simulazione (discreta). Infatti assai spesso nei blocchi si ha una dipendenza istantanea tra ingresso e uscita. Quando si collegano due blocchi di questo tipo in feedback, ne risulta un sistema di equazion iintegro-differenziali. Nella simulazione si dice che risulta un anello di calcolo senza ritardo. Una soluzione consiste nel risolvere per ogni istante il sistema di equazioni. Questo pero' rendi il calcolo molto complesso. L'altra soluzione invece consiste nell'introdurre opportuni ritardi in modo da rompere gli anelli senza ritardo e quindi rendere le equazioni direttamente computabili. Cio' vuol dire che si utilizzeranno in alcuni punti i valori in istanti precedenti. Spesso questo e' fatto in maniera implicita quando si discretizzano le euazioni continue, scegliendo oppurtunamente lo schema di discretizzazione. Questo approccio puo' introdurre inconsistenze, per cui deve poi essere effettivamente valutato nei casi specifici. Nell'introdurre il nostro schema generale di interazione ci eravamo posti come obbiettivo di esplicitare nel blocco di interazione questo tipo di problemi.

Allo scopo di definire strutture direttamente computabili esaminiamo ora in particolare il blocco dell'eccitatore. Nello schema generale di interazione da noi proposto l'eccitatore è definito come un sistema dinamico, causale, generalmente non lineare e tempo variante. Per caratterizzare meglio l'eccitatore noi proponiamo una struttura costituita da una parte lineare che tiene conto della memoria del sistema ed da una parte non lineare istantanea (Fig. 1). Le relazioni di ingresso ed uscita sono

$$X(n+1) = F_L\,[X(n),\ U(n),\ U_E(n),\ Y_E(n)]$$

per la parte lineare, dove $X(n)$ vettore di stato, $U_E(n)$ e $Y_E(n)$ vettori degli ingressi ed uscite dell'eccitatore verso il blocco di interazione, $U(n)$ vettore degli ingressi applicati dall'esterno dell'algoritmo, ad esempio da parte dell'esecutore. Per la parte non lineare

$$Y_E(n) = F_{NL}[X(n),\ U(n),\ U_E(n)]$$

dove F_{NL} è una funzione vettoriale istantanea generalmente non lineare.

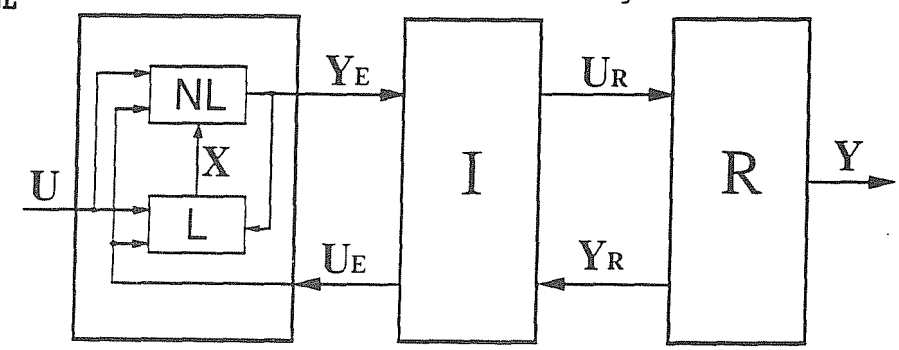


Fig. 1

E' importante notare una caratteristica essenziale nella definizione di questa classe, comune a quasi tutti gli algoritmi basati sui modelli fisici, che è la presenza del vettore Y_E dell'uscita dall'eccitatore fra gli ingressi della parte lineare. Ovviamente Y_E non può essere anche ingresso alla parte non lineare per ragioni di computabilità. La parte lineare funge quindi da separatore per eliminare anelli di calcolo senza ritardo. La motivazione di questa proprietà può essere fatta risalire all'uso di coppie di grandezze estensive/intesive nelle descrizioni delle proprietà degli elementi fisici e quindi dei modelli basati sulla fisica.
Anche se questa struttura di eccitatore e' meno generale del sistema dinamico discreto non lineare, esso copre molti casi interessanti. Per esempio, in molti eccitatori fisici la non linearità sta nelle caratteristiche elastiche delle sue parti. Una buona descrizione di tali caratteristiche generalmente prevede una relazione di ingresso uscita istantanea o con isteresi, che viene realizzata dalla parte non lineare del sistema, sfruttando eventualmente il feedback dall'uscita della non linearità per l'isteresi. L'inerzia dell'elasticità può essere descritta dalle parti lineari dell'eccitatore.

3. Realizzazione degli eccitatori fisici

Vediamo ora come la classe di algoritmi proposta si presti per realizzare eccitatori fisici. Normalmente gli eccitatori fisici possono essere divisi in due categorie: "eccitatori meccanici" (come ad esempio martelletto, plettro, archetto, bacchetta ecc..) e "eccitatori fluodinamici" (ancia, labbra, glottide ecc..). Gli eccitatori meccanici spesso sono convenientemente descritti come reagenti con forza ad ingressi di tipo spostamenti o velocità. La presenza di masse viene descritta dalla parte lineare dello schema proposto.
Un primo esempio è la corda percorsa dal martelletto (Borin&Sarti 1989). Il modello del martelletto ha una massa, che rappresenta il capo del martelletto, e una molla non lineare con lunghezza zero a riposo, che descrive la caratteristica di compressione del feltro. Le espressioni risultanti sono:

$$m\, y = f_o - f$$

$$f = F_{NL}\,(y - s)$$

dove f_o forza applicata dall'esecutore, f la forza di reazione della corda, y la posizione del martelletto e s posizione della corda nel punto di contatto. L'elasticità della molla viene descritta da F_{NL} che può essere, ad esempio, $F_{NL}\,(n) = c\, h^4$ per $k>0$ e $F_{NL}\,(h) = 0$ per $h<0$. dove h è l'allungamento della molla dalla sua posizione di riposo.
Sostituendo alla derivata la differenza, chiamato T il quanto temporale , si ha

$$y\,(n+1) = 2y(n) - y(n-1) + (f_o - f(n)) \cdot \frac{T^2}{M}$$

$$f(n) = F_{NL}(y(n) - s(n))$$

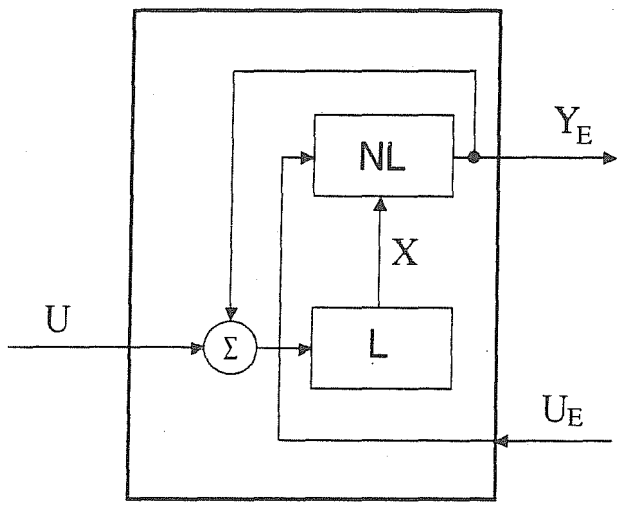


Fig. 2

Queste equazioni corrispondono ad un sistema del tipo proposto (Fig. 2), con le seguenti posizioni:

$X = [y(n), y(n-1)]$, $U = f_o$, $U_E = s$, $Y_E = f$.

Il caso della corda pizzicata condivide con il precedente lo stesso risuonatore. Per mostrare la proprietà di modularità e versatilità del metodo, presentiamo due modelli diversi di eccitatore. Nel primo, il plettro è modellato come una semplice variazione del modello del martelletto. Infatti per tener conto della flessione del plettro e del successivo rilascio della corda, è sufficiente assicurare che la forza di interazione diventi zero quando l'allungamento della molla diventa maggiore di un valore h_m (Borin et al. 1990b). Questo può essere raggiunto con una differente caratteristica di elasticità della molla

$F_{NL}(h) = c\ h^4$ per $0 \leq h_m$ e $F_{NL}(h) = 0$ altrove.

A parte questa differente funzione non lineare, il sistema ha la stessa struttura e percio' valgono le stesse considerazioni di prima.

Un modello più sofisticato del precedente si può ottenere notando che nel caso della corda pizzicata, l'ipotesi di eccitazione monodimensionale appare riduttiva. Si può allora fare riferimento ad un modello del plettro bidimensionale, capace cioè di eccitare il risuonatore secondo due componenti ortogonali e disaccoppiate, così come rappresentato in Fig. 3. In tale modello si ipotizza che il plettro sia perfettamente rigido e privo di massa; la massa m dà conto dell'inerzia della mano dell'esecutore, e la molla non lineare K dà conto della comprimibilità dei polpastrelli. Il plettro è quindi tenuto in modo da potersi inclinare, ma non flettere, sotto la sollecitazione della corda. Con riferimento alla Fig. 3, si supponga che la forza esercitata dalla molla sulla massa agisca solo lungo la direzione y. In questo caso è semplice trovare le componenti di tale forza lungo z e lungo la

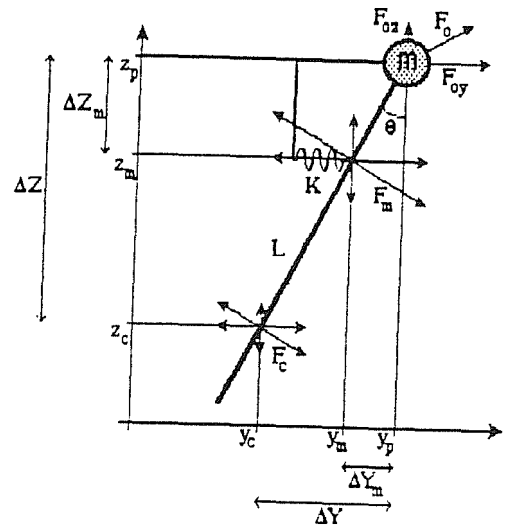


Fig. 3

normale al punto di eccitazione. Essendo infatti per ipotesi:

$$F_{my} = K(\beta\Delta y)^{\alpha}$$

si prova facilmente che è:

$$F_m = \frac{K(\beta\Delta y)^{\alpha}}{\cos(\theta)}$$

$$F_{my} = \{K(\beta\Delta y)^{\alpha}\}\ \frac{\text{sen}(\theta)}{\cos(\theta)} = \{K(\beta\Delta y)^{\alpha}\}\text{tg}(\theta) = \{K(\beta\Delta y)^{\alpha}\}\ \frac{\Delta y}{\Delta z}$$

dove $\theta = \text{atan}(\Delta y/\Delta z)$, e dove $\beta = \Delta z_m/\Delta z$; β indica così il rapporto di proporzionalità della leva costituita dal plettro. Utilizzando le classiche equazioni della dinamica, le relazioni che regolano l'evoluzione del sistema divengono:

$$m\ddot{y}_p = F_{ey} - F_{my} = F_{ey} - K(\Delta y)^{\alpha}\left(\frac{\Delta z_m}{\Delta z}\right)^{\alpha+1}$$

$$m\ddot{z}_p = F_{ez} - F_{mz} = F_{ez} - K(\Delta y)^{\alpha}\left(\frac{\Delta y}{\Delta z}\right)\left(\frac{\Delta z_m}{\Delta z}\right)^{\alpha+1}$$

Per completare il modello dell'eccitatore occorre discutere ora le condizioni di contatto. Nella corda percossa, la condizione di contatto andrebbe calcolata mediante un sistema dinamico non lineare, la cui uscita va ad influenzare direttamente la funzione istantanea non lineare preposta la calcolo della forza. In tale situazione ne consegue che la parte non lineare complessiva dell'eccitatore possiede una dinamica propria, il che e' contrario alla filosofia dello schema proposto. E' tuttavia possibile riscrivere le equazioni del sistema di aggiornamento del contatto in modo che esso operi aggiornando il contatto presente sulla base delle reazioni del risuonatore U_E, dello stato X e della conoscenza delle condizioni di contatto precedente. In tale situazione, entra a far parte dello stato della parte lineare, riconducendo cosi' il calcolo della forza e del nuovo contatto ad una funzione non lineare istantanea. Dopo la riscrittura esposta il modello ritorna simile allo schema del martelletto di Fig. 2, fermo restando che in questo caso posizione e forza sono variabili vettoriali con l'aggiunta di una ulteriore variabile di stato interna, la condizione di contatto, la cui dinamica e' assicurata dalla memoria contenuta nella parte lineare e il cui aggiornamento e' operato dalla parte non lineare. Tale schema e' mostrato in Fig. 4 dove e' indicata con la linea tratteggiata questa nuova variabile.

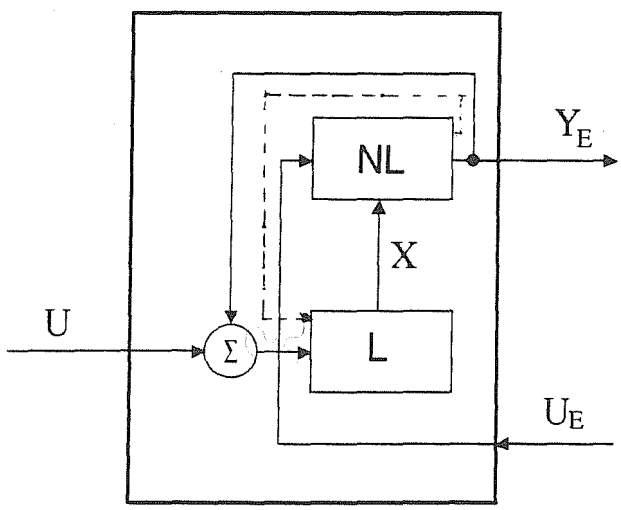


Fig. 4

4. Eccitatore fluidodinamico

Un'altro tipo fondamentale di eccitatore e' quello fluidodinamico. Come esempio di applicazione del metodo consideriamo il caso del clarinetto. Il flusso u(t) che dall'eccitatore va al risuonatore dipende essenzialmente dalla differenza di pressione, tra la bocca del musicista $p_o(t)$ e l'inizio della canna p(t), e dalla dinamica dell'ancia. La

descrizione accurata della dinamica dell'ancia è molto complicata. In letteratura si trovano alcune soluzioni approssimate. In quella più classica si suppone che la dinamica dell'ancia sia descrivibile attraverso una equazione differenziale di tipo

$$\ddot{y} + g\,\dot{y} + \varpi^2 y = (p - p_o)/\mu$$

dove y(t) rappresenta la deflessione dell'ancia dalla sua posizione di equilibrio, ϖ è la frequenza naturale di oscillazione dell'ancia, g è il coefficiente di smorzamento dell'ancia causato dall'attrito viscoso con l'aria e dalla rigidezza dell'ancia stessa, e μ è il rapporto tra massa e superficie (efficace) dell'ancia.
Il flusso uscente dall'eccitatore verso il risuonatore e' dato da

$$u = a(y + h)^{4/3}\,(p_o - p)^{2/3}\,1(y - 1) - S\,\dot{y}$$

dove a è una costante empirica , h è l'apertura dell'ancia a riposo, 1(.) e' la funzione di Heaviside, per definire il supporto della funzione, e S è la superficie effettiva dell'ancia. Il secondo addendo tiene conto della corrente acustica prodotta dal movimento dell'ancia. Discretizzando abbiamo

$$y(n + 1) = -b_o\,y(n) - b_1\,y(n-1) + a_1(P - P_o)$$

$$u(n) = a(y(n)+h)^{4/3}\,(p_o(n)-p(n))^{2/3}\,1(y(n)-h) - \frac{S}{T}\,[y(n) - y(n-1)]$$

dove b_0, b_1, e a_1 sono delle opportune costanti. Da questa espressione risulta che, ponendo X = [y(n), y(n-1)], U=p_0, U_E=p, Y_E=u abbiamo il modello di Fig. 5a tipico per gli eccitatori fluidodinamici.

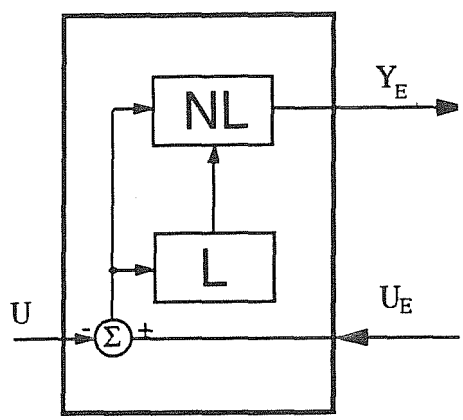


Fig 5a

Nel modello più semplificato del clarinetto, proposto da McIntyre et al. (1983), viene trascurata la dinamica dell'ancia. Pertanto y è proporzionale a $(p-p_o)$ e l'eccitatore è costituito solo dalla funzione non lineare.

Si può osservare inoltre che se consideriamo in analogia a McIntyre et al.(1983) come ingresso all'eccitatore $U_E = p_h = p-Z_o u$, otteniamo un sistema che in generale non è più del tipo da noi proposto. Infatti l'uscita Y_E non è più disaccoppiata dall'ingresso della parte non lineare mediante lo stato della parte lineare. Risulterebbe infatti che

$$Y_E = F_{NL}\,(X(n),\ U,\ U_E,\ Y_E)$$

Questo richiede quindi una rielaborazione delle equazioni e la soluzione di un sistema lineare (Schumacher 1981) per ogni valore dell'uscita da calcolare. Se si aggiunge un ritardo si può ricondurre anche questo caso alla struttura qui proposta considerando come stato X = [y(n), y(n-1), u(n-1)], aggiungendo cioe' il flusso allo stato. Risulta ancora un algoritmo descrivibile con lo schema di Fig. 5a, con però differente significato per i vettori di ingresso, uscita e stato.
Se invece si trascura la dinamica dell'ancia, si può calcolare direttamente l'uscita esplicitando l'equazione precedente (Balena&DePoli 1985, Smith 1986) ricavando cioe' Y_E in funzione di U e U_E. Si ottiene cosi'

$$Y_E(n) = F_{NL}[U(n),\ U_E(n)]$$

In questo caso (Fig. 5b) non occorre piu' la parte lineare e l'evoluzione dello stato.

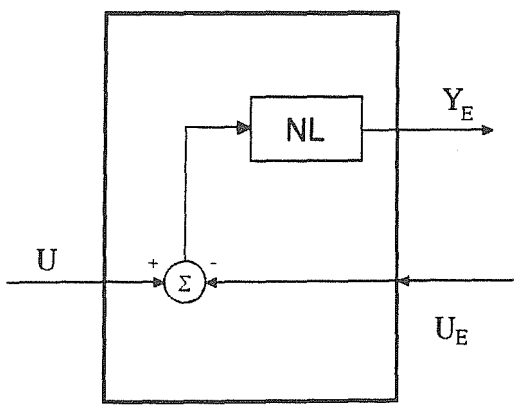


Fig 5b.

5. Eccitatori pseudofisici: modello caotico

Si descrive ora un possibile modello di eccitatore generalizzato, basato su un sistema retto da equazioni fisiche, ma privo di corrispondenza con qualsiasi strumento naturale. Si vuole realizzare un modello in grado di produrre segnali interpretabili come eccitazioni. Data la totale libertà interpretativa in cui ci si muove, non è necessario considerare i segnali prodotti come forze, pressioni o altro: semplicemente, si potrà fare riferimento ad essi come ad una sequenza di numeri, che potrà avere interesse se interpretata come segnale musicale. Per ragioni essenzialmente pratiche, si privilegieranno eccitatori in grado di produrre segnali riccamente strutturali. A tale proposito faremo riferimento ora ad un sistema caotico, in grado cioè di produrre un segnale in uscita dotato di caratteristiche di elevata impredicibilità e varietà. Si consideri un sistema dinamico che descrive l'evoluzione in uno spazio bidimensionale di tre masse puntiformi, collegate tra loro da molle lineari, uguali e di lunghezza a riposo non nulla L. Il sistema è descritto dalle relazioni

$$m_i \, X_i = F_i \qquad \text{con } i = 1,2,3$$

dove X_i è il vettore delle coordinate del baricentro della massa i, e F_i è il vettore della forza agente sulla massa i; tale forza ha la seguente espressione:

$$F_i = F_{oi} - K \, (d(X_i,X_j)-L)u_{ij} - K \, (d(X_i,X_k)-L)u_{ki}$$

dove i pedici j e k si riferiscono alle due masse contigue, K è la costante elastica della molla, u_{ij} è il versore diretto dalla massa i alla massa j e d(.,.) è la distanza euclidea tra i baricentri considerati. F_{oi} è la forza esterna applicata alla i-esima massa. E' facile infine provare che le componenti F_{ijx} e F_{ijy} della forza generica F_{ij} tra due masse sono date da:

$$F_{ijx} = -K \frac{H-L}{H} (X_i - X_j)$$

$$F_{ijy} = -K \frac{H-L}{H} (y_i - y_j)$$

dove si è posto:

$$H = (X_i - X_j)^2 + (y_i - y_j)^2$$

In assenza di forze esterne, il sistema è posto in movimento perturbando le masse dalla loro posizione di equilibrio. L'evoluzione del sistema è assai complessa, ed è del tutto imprevedibile. Inoltre essa presenta una spiccata tendenza all'instabilità delle traiettorie: una piccola deviazione nel valore dei parametri iniziali porta a traiettorie completamente differerenti nello spazio di stato. Si noti tuttavia che il sistema non è di per sè stesso divergente. Il sistema così descritto può quindi essere considerato un "generatore di segnali caoti-

ci"; esso può essere utilizzato come produttore di suono a sè stante. E' tuttavia necessario osservare che non è possibile ottenere un controllo sui parametri musicalmente interessanti di un tale generatore; l'ampiezza e la frequenza istantanee del segnale prodotto non dipendono direttamente da alcuno dei parametri del sistema; inoltre, tali grandezze sfuggono completamente a qualsiasi tentativo di caratterizzazione e previsione. Diventa quindi necessario l'impiego di un sistema esterno, che provveda in qualche misura al controllo di tali parametri. Come risuonatore usiamo una semplice linea di ritardo che riceve l'azione dell'eccitatore al suo ingresso e fornisce una "reazione" data dal semplice ingresso ritardato e pesato da un opportuno coefficiente. Per la connessione del modello di eccitatore con il risuonatore qui è opportuno utilizzare come grandezza in uscita del sistema il valore della forza agente su una delle masse per opera delle rimanenti. Tale grandezza e' rappresentata da un vettore bidimensionale; essa è quindi inviata ad una coppia di linee di ritardo, ciascuna delle quali propaga una componente del segnale in uscita. Tale forza è immessa nelle linee di ritardo, e viene ritornata al punto di partenza dopo un tempo nT; la forza di ritorno agisce sulla massa in modo del tutto analogo a quello delle forze interne del sistema. Quando c'è il risuonatore, dopo un transitorio nel quale il sistema cerca una soluzione di compromesso tra il comportamento caotico dell'eccitatore e il modo oscillatorio periodico del risuonatore, si giunge ad una situazione di equilibrio in cui predomina il comportamento periodico. Il modello descritto e' ancora della classe proposta. Infatti ora abbiamo (Fig. 6) che

$U = [\ F_{o1},\ F_{o2},\ F_{o3}\ ]$ $\quad U_E = F_{r1}$ $\quad Y_E = [\ F_1,\ F_2,\ F_3]$

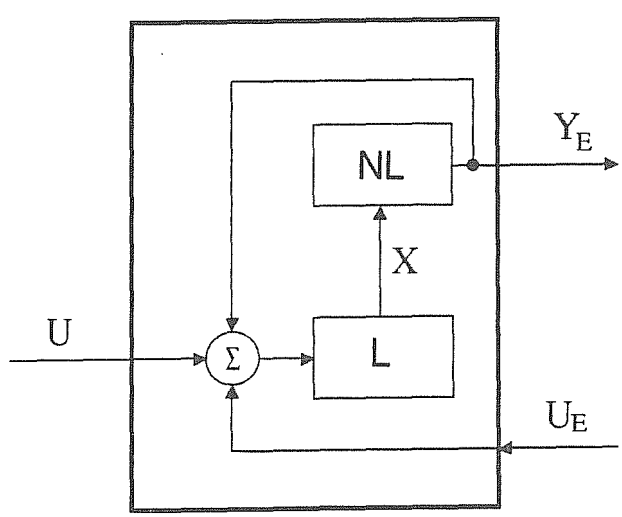


Fig. 6

6. Eccitatori non fisici

Quando si rimuove la condizione di usare coppie di variabili fisiche estensive e intensive, come ad esempio tensione e corrente o pressione e flusso, si ottiene la struttura più generale di interazione, che è quella non fisica. Presentiamo qui un'esempio fra i più semplici che possono essere derivati dalla classe da noi preposto. Consideriamo infatti come eccitatore il classico operatore della modulazione di frequenza. Esso calcola la seguente espressione

$$y(n) = \text{sen}\ (2\pi nf/f_S + \phi(n))$$

dove $\phi(n)$ è il segnale modulante, f_S è la frequenza di campionamento. Nel nostro caso $\phi(n)$ è preso come ingresso dell'eccitatore U_E. La struttura risultante è composta dalla parte lineare

$$x(n+1) = x(n)+S$$

dove $S = 2\pi f/f_S$ e dalla parte non lineare

$$y(n) = \text{sen}\ (x(n)+\phi(n))$$

Se facciamo la posizione $X = [x(n)]$, $U = S$, $U_E = \phi$, $Y_E = y$, otteniamo

$$X(n+1) = X(n+1) + U(n)$$

$$Y_E(n) = \text{sen}\ [X(n) + U_E(n)]$$

illustrato in Fig. 7.

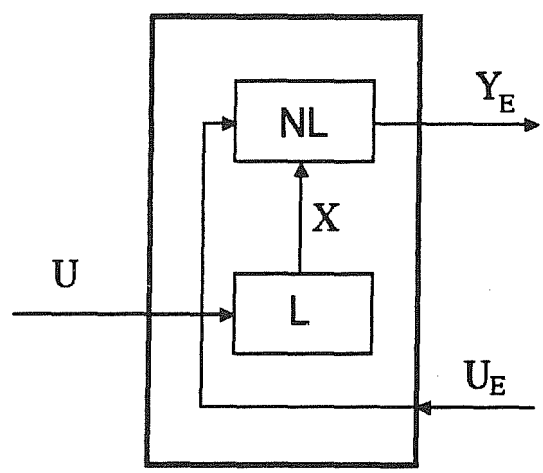


Fig. 7

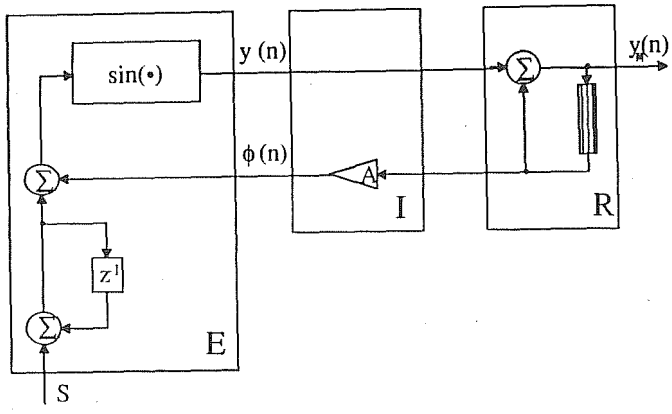


Fig. 8

Se interconnettiamo questo eccitatore con un risuonatore costituito da una linea di ritardo chiusa otteniamo una FM con risuonatore interagente (Fig. 8). Si può osservare l'analogia strutturale di questo algoritmo con lo schema base degli oscillatori musicali proposto da McIntyre et al.(1983). Nel blocco di interazione si può porre un fattore di riscalamento A analogo all'indice di modulazione nella modulazione di frequenza classica. Quando A assume valori piccoli, controlla l'allargamento dello spettro; quando assume valori elevati il suono diventa caotico e quindi molto rumoroso. Per valori intermedi esso controlla il sorgere di granulosita' e discontinuita', dovuti al fatto che la fase istantanea non e' piu' monotona. Dal punto di vista percettivo il suono risulta sempre piu' frammentato. La complessita' computazionale e' simile a quella della semplice FM statica. Questa struttura consente pero' un'evoluzione dinamica del timbro. Questo esempio dimostra come la classe proposta consenta di generalizzare facilmente gli algoritmi tradizionali.

7. Conclusioni

Nella musica tradizionale, lo strumento ha una duplice funzione. Infatti, oltre ad essere il mezzo con cui viene prodotto il suono, puo' anche essere considerato come astrazione di una classe di suoni, caratterizzati da un timbro, un comportamento dinamico e da certe possibilita' espressive. Noi crediamo che questa duplice funzione sia vera in generale e possa essere anche applicata agli strumenti sintetici. In particolare negli strumenti numerici il mezzo di generazione e' descritto da un algoritmo. Noi crediamo quindi che la struttura dell'algoritmo di sintesi contribuisca a definire una classe di suoni anche indipendentemente dalle interpretazioni che hanno motivato l'algoritmo stesso. Per questa ragione cerchiamo di esplorare le strutture. In particolare in questo lavoro abbiamo definito una classe di strutture piuttosto generale per gli eccitatori da inserire in uno schema in cui essi interagiscano con dei risuonatori. I singoli algoritmi possono essere ricavati sia da modelli fisici che da strutture piu' astratte, che possono trarre ispirazione dalla realta' quando utile, ma che sono piuttosto generatrici di una loro realta' propria.

Riferimenti:

J.M. Adrien, "Physical model synthesis: the missing link", G. De Poli, A. Piccialli, and C. Roads, ed., **Representations of Musical Signals**, Cambridge MA, MIT Press, 1991, pp. 269-297.

F. Balena, G. De Poli, "Un modello semplificato del clarinetto mediante oscillatore non lineare", **Atti del VIII Colloquio di Informatica Musicale**, 1985, pp. 111-138.

G. Borin, G. De Poli, and A. Sarti, "A modular approach to resonator-excitator interaction in physical models synthesis." **Proc. International Computer Music Conference**, 1989, pp. 46-50.

G. Borin, G. De Poli, and A. Sarti, "Generalizing the physical model timbre clars", **Proc. Colloquium on Physical Models**, Grenoble, ACROE, 1990a.

G. Borin, G. De Poli, and A. Sarti, "Excitator-resonator interaction and sound synthesis." **Proc. Colloquium on Physical Models**, Grenoble, ACROE, 1990b.

G. Borin, and A. Sarti, "Interazione martelletto corda nella sintesi del pianoforte", **Atti del VIII Colloquio di Informatica Musicale**, Cagliari, 1989, pp. 9-16.

J.L. Florens, "Autour de la simulation instrumentale modulaaire et du controle gestuel", **Proc. Colloquium on Physical Models**, Grenoble, ACROE, 1990.

J.L. Florens, and C. Cadoz, "The physical model: Modeling and simulating the instrumental universe, in G. De Poli, A. Piccialli, and C. Roads, eds., **Representations of Musical Signals**, Cambridge MA, MIT Press, 1991, pp. 227-268.

M.E. Mc Intyre, R.T. Schumacher, and J. Woodhouse, "On the oscillations of musical instruments". **Journal of the Acoustical Society of America**, Vol. 74, No. 5, 1983, pp. 1325-1345.

R.T. Schumacher, "Ab initio calculations of the oscillations of a clarinet", **Acustica**, Vol. 48, No. 2, 1981, pp. 71-85.

J.O. Smith, "Efficient simulation of reed-bore and bow-string mechanism", **Proc. International Computer Music Conference**, 1986, pp. 275-280.

METODI DI ANALISI DELLE MICROVARIAZIONI DI AMPIEZZA E FREQUENZA DEI SUONI MUSICALI E LORO APPLICAZIONE ALLO STUDIO DI UN ORGANO BAROCCO

Giovanni B. Debiasi, Giovanni Spagiari

D.E.I.- C.S.C.- Universita' di Padova
Via Gradenigo 6a, 35131 PADOVA
Tel +39 -49 -8287500 Fax +39 -49 -8287699

Keywords: Musical Sounds Timbre Evaluation, Transients Analysis, Small Fluctuations Tracking, Pipe Organ Acoustic Description.

ABSTRACT

Musical instruments produce nearly periodic waveforms, whose spectral components fluctuate both in the amplitude and in the frequency. We call "microvariations" these fluctuations. Their analysis can be done in several ways, that often may be regarded as complementary methods. On this paper we review some of these methods and we explain some improvements that allow a very fine detection of the microfluctuations, whose knowledge is an important key for an accurate and natural synthesis of the musical sounds.

Our methods of analysis, after several laboratory tests, have been used for an extensive investigation on the acoustic characteristics of a valuable baroque pipe organ. A very large amount of data have been evaluated in a rather short time, giving a complete acoustic description of the organ and suggesting very interesting hypotheses about the charm of its sounds.

1 - INTRODUZIONE

I suoni musicali, prodotti da strumenti tradizionali o dalla voce, sono caratterizzati da fluttuazioni, non necessariamente del tutto casuali, della frequenza e dell'ampiezza delle varie componenti spettrali, che si traducono in fluttuazioni nella forma e nella durata del periodo delle successive forme d'onda. Questo insieme di fenomeni,abbastanza trascurato per il passato, sta ora riscuotendo sempre maggiore interesse, specie in vista di poter sintetizzare con maggiore naturalezza i suoni degli strumenti tradizionali o della voce: si vedano, per esempio, le recenti pubblicazioni [1], [2], [3].

Nel presente lavoro le fluttuazioni sopra ricordate vengono indicate con il nome di "microvariazioni", allo scopo di distinguerle da quelle intenzionalmente imposte per ottenere effetti di tremolo, vibrato, portamento o simili. Il rilievo e l'evidenziazione delle microvariazioni possono essere eseguiti con varie tecniche, delle quali quelle numeriche

presentano grande interesse per la possibilita` di automatizzare largamente le varie operazioni.

Molto suggestivo appare il metodo proposto da Schumacher e Chafe in [2], che consente una visualizzazione bidimensionale a colori della struttura di una forma d'onda, permettendo valutazioni qualitative circa le differenze tra forme d'onda successive o di differenti strumenti. Esso, tuttavia, si presta a qualche critica per la arbitrarieta` con cui vengono definiti i successivi intervalli di suono sottoposti ad analisi digitale sincrona di Fourier, ottenuta ricampionando (via software) ogni intervallo con un numero intero eguale di campioni. Analoghi risultati qualitativi, oltre che quantitativi, si possono ottenere con la tecnica descritta nel lavoro di Debiasi e Dal Sasso [4], che si presta bene ad evidenziare in termini di timbro e di complessita` non solo le caratteristiche dei suoni in regime stazionario, ma anche la loro evoluzione nel tempo. Nella presentazione di tale tecnica effettuata in [4], si era messo in evidenza come essa consenta una rapida ed agevole valutazione automatica delle caratteristiche complessive di un registro di organo a canne (ed eventualmente di ogni altro strumento a suono sostenuto), in termini di due grandezze definite a tale scopo: timbro @ e complessita` C. Esse definiscono rispettivamente un angolo di rotazione ed una distanza radiale su un diagramma polare e consentono, in tale modo, di collocare il punto corrispondente al suono a regime di un dato registro o strumento in una zona del diagramma polare caratteristica della famiglia di strumenti a cui il registro o lo strumento in esame appartiene.

2 - SVILUPPI DEL METODO DEL DIAGRAMMA POLARE

Il metodo del diagramma polare sopra ricordato e` stato applicato anche allo studio delle microvariazioni del timbro e della complessita` dei suoni durante il transitorio di attacco, e la sensibilita` della tecnica adottata ha confermato la sua attitudine a scoprire in quale modo evolvono in transitorio le caratteristiche acustiche dei suoni, consentendo di dare ad esse attributi qualitativi, oltre che quantitativi, di tipo analogo al giudizio che possono esprimere i gruppi di ascolto tradizionalmente impiegati per la valutazione dei suoni a regime. Ovviamente tali gruppi trovano molta difficolta` ad esprimere un giudizio per quanto riguarda il transitorio di attacco, data la sua brevissima durata; invece non si hanno problemi con la tecnica sopra illustrata. Per effettuare le analisi in transitorio e` sufficiente isolare, dal file del suono digitalizzato, intervalli che comprendano ciascuno non meno di 5 periodi di segnale o non meno di 20 ms di durata.

I campioni di ciascun intervallo vengono pesati con una opportuna finestra (per es.di Hamming) e sottoposti a trasformata digitale di Fourier (DFT), e si eseguono tali operazioni su intervalli spaziati di circa 10 ms l'uno dall'altro. Si ottiene una matrice di dati, nelle cui righe si trovano le informazioni relative a ciascuna componente spettrale del suono ogni 10 ms successivi e nelle cui colonne sono le informazioni relative allo spettro. La insensibilita` dell'orecchio a variazioni timbriche in intervalli di tempo inferiori alla decina di millisecondi e la pesatura compiuta dalla finestra adottata consentono di ritenere che i dati della matrice siano rappresentativi dell'evoluzione spettrale del suono percepito, indipendentemente dall'arbitrarieta` degli intervalli scelti. Dai dati della matrice si puo` ricavare l'evoluzione temporale del timbro @ e della complessita` C dei suoni, rappresentandoli con diagrammi a righe (una ogni 10 o multipli di 10 ms, v. figure 1a e 1b), o con un insieme di punti su un diagramma polare del tipo di quello illustrato in [4] e qui riprodotto in fig. 1c.

Questo metodo di analisi sembra molto adatto ad evidenziare in forma sintetica, mediante un sistema automatico operante su PC, le microvariazioni di un suono in regime transitorio o stazionario. Tuttavia non consente di valutare le microvariazioni delle sue

componenti spettrali in maniera abbastanza fine da consentirne un ulteriore studio a scopo di risintesi. Uno strumento adatto a questo scopo e` dato dal filtraggio digitale eterodina, che e` stato messo a punto nel contesto del presente lavoro, apportando ad esso alcuni miglioramenti adatti a renderne l'impiego agevole e preciso.

3 - METODO DI FILTRAGGIO ETERODINA

Il metodo di filtraggio eterodina e` ben noto nel campo della elettronica analogica ed annovera innumerevoli impieghi, spesso di antichissima data, che spaziano dalla selezione delle portanti radio nei radioricevitori, alla selezione delle componenti spettrali negli analizzatori di spettro analogici e ad altre svariatissime applicazioni, specie nel settore delle misure elettroniche. Esso si basa, essenzialmente, su una conversione di frequenza che porta le componenti spettrali, che interessa analizzare, entro la banda di un filtro dotato di caratteristiche prefissate, particolarmente accurate e precise. La versione digitale del filtraggio eterodina e` stata studiata da numerosi autori, come per esempio si puo` vedere in [5], [6], [7], [8].

Nella presente applicazione, lo spostamento di frequenza delle componenti spettrali, che si vogliono analizzare, viene compiuto in modo da trasportarle nell'intorno della frequenza zero: in tale modo ogni scostamento dalla frequenza nominale viene rappresentato con uno scostamento da zero e, dilatando opportunamente la scala delle frequenze, diviene possibile valutare scostamenti comunque piccoli, rendendo il metodo di analisi sensibile ed accurato quanto necessario. Inoltre, a differenza dei metodi numerici proposti in precedenza, atti a fornire la fase delle frequenze trasposte, con evidenti problemi di discontinuita` periodica, il metodo che qui si propone calcola la differenza di fase tra un campione e quello precedente, consentendo di ricavare immediatamente la derivata della fase, che fornisce, campione per campione, la frequenza istantanea trasposta. Gli algoritmi e le procedure di calcolo, attuati su PC, vengono riassunti in Appendice a questo lavoro. Essi consentono di ottenere direttamente, come gia` accennato, per ciascuna componente spettrale del suono in esame, gli scostamenti dal valore nominale della frequenza e dell'ampiezza con precisione molto elevata. La fig. 2 mostra, per esempio, i risultati dell'analisi di una sinusoide a 256 Hz modulata in frequenza con deviazione di 4 Hz ed in ampiezza con profondita` del 20%, con modulante sinusoidale a 21 Hz. Si vede come il metodo consenta di valutare agevolmente deviazioni di ampiezza anche inferiori al percento, deviazioni di frequenza di frazioni di Hz e di evidenziare frequenze modulanti sino ad alcune decine di Hz. La precisione con cui e` possibile determinare gli scostamenti di frequenza da quella nominale, consente, tra l'altro, di misurare eventuali inarmonicita` delle parziali. Tutto cio` si puo` ottenere sia in transitorio di attacco, sia a regime, integrando cosi` con precisi dati analitici le valutazioni di tipo globale dell'altro metodo sopra ricordato.

4 - STUDIO DI UN ORGANO BAROCCO

Questi due metodi di indagine sono stati utilizzati estensivamente per rilevare le microfluttuazioni dei suoni di un pregiato organo a canne dell'epoca barocca esistente in localita` Gambarare, in provincia di Venezia. Si tratta di una monumentale opera costruita dal celebre organaro veneto Gaetano Callido tra il 1786 e il 1798, recentemente restaurata a cura della Ditta Piccinelli di Padova, e quindi perfettamente funzionante. Costituisce, a giudizio degli esperti, un notevolissimo esempio dell'arte organaria barocca e, data la relativa facilita` di accesso, tale organo e` stato scelto per effettuare una prima serie estensiva di rilievi e di analisi delle microvariazioni dei suoni

prodotti dai vari registri disponibili. I risultati di tali analisi, tuttora in corso di approfondimento, si possono rivelare utili sia al fine di ampliare le conoscenze relative al timbro dei registri e all'impostazione fonica dell'intero strumento, sia al fine di migliorare la riproduzione digitale dei suoni destinati ad organi elettronici di raffinate prestazioni acustiche.

La registrazione dei suoni ha comportato la soluzione di non lievi problemi di ordine pratico: essa e' stata effettuata di notte, con un microfono direzionale a risposta piatta, allo scopo di minimizzare le interferenze date dal rumore ambientale e dalle riverberazioni, ineliminabili nella chiesa dove l'organo e' collocato. Sono stati ripresi e digitalizzati i suoni di tutte le note dei registri o delle combinazioni di cui all'elenco seguente.

Principale 8' bassi + Principale 8' soprani
Ottava 4'
Contrabbasso 16' + Ottava di contrabbasso 8' (pomelli stabilmente uniti)
Contrabbasso 16' + Ottava di contrabbasso 8' + Principale 8' bassi
Trombone 8' (ancia)
Voce umana 8' soprani + Principale 8' soprani
Flauto 4' bassi + Flauto 4' soprani
Fagotto 8' bassi (ancia) + Tromba dolce 8' soprani (ancia)
Tromboncini 8' bassi (ancia) + Tromboncini 8' soprani (ancia)
Ripieno (Principale 8' + Ottava 4'+ XV 2' + XIX 1'1/3 + XXII 1' + XXVI 2/3'
+ XXIX 1/2' + XXXIII 1/3'+ XXXVI 1/4')

L'analisi, effettuata successivamente, e' stata limitata, per ora, a 2 note per ottava, precisamente DO e FA# distanti tra loro 600 cent (mezza ottava). Sono state eseguite valutazioni delle microvariazioni di timbro e brillantezza in transitorio di attacco ed in regime stazionario e sono state poste in evidenza le microvariazioni di frequenza e di ampiezza delle prime 6 parziali, che determinano in modo sostanziale il timbro.

Le figure 1 a, b, c, gia' considerate in precedenza, sono relative alla nota DO4 del Principale 8' soprani e mettono in evidenza, per esempio, l'evoluzione del timbro @ e della complessita' C nel transitorio di attacco: la numerazione delle righe (in ascisse) e dei punti (nel diagramma polare) si riferisce a successivi intervalli di analisi spaziati di 20 ms. Si puo' osservare che, in transitorio, il timbro evolve rapidamente dalla regione tipica delle viole a quella dei principali e la complessita' si sposta da valori inizialmente elevati (C=54) a valori inferiori a 20, che rispecchiano la scarsa ricchezza di parziali dei principali.

A regime i punti del diagramma polare si addensano attorno al punto n.6; pero' il diagramma di @ mostra ancora sensibili variazioni di timbro tra 240 e 300 ms dall'attacco (rilevabili pure sul diagramma polare come "stiramento" del "cluster" di punti). In fine, per quanto concerne le microvariazioni di ampiezza e frequenza delle 6 prime parziali, omettendo le figure ad esse relative, si puo' riferire che la fondamentale appare molto stabile in ampiezza (variazioni dell'ordine del percento) e in frequenza (scostamenti mediamente inferiori a 0,5 Hz), mentre le parziali successive presentano fluttuazioni superiori, con un massimo per la 4^ parziale, che presenta fluttuazioni di ampiezza sino all'80% e di frequenza sino oltre 7 Hz. Nella seguente tabella sono in fine riportati gli scostamenti dai valori nominali misurati per le frequenze delle prime 6 parziali: come si vede essi sono quasi proporzionali al numero d'ordine delle parziali stesse (con una leggera tendenza all'aumento per le ultime due parziali) e sembrano mettere in evidenza che si tratta non tanto di inarmonicita' delle parziali, quanto di una accordatura lievemente crescente della canna; fa solo eccezione la 4^ parziale: l'anomalia del suo scostamento

negativo si puo` ricondurre alla forte instabilita` di frequenza riscontrata e puo` forse venire interpretata come inarmonicita` vera e propria.

n. della parziale	frequenza nominale (Hz)	scostamento (Hz)
1	262	+ 0,068
2	524	+ 0,123
3	986	+ 0,188
4	1048	- 0,647
5	1310	+ 0,374
6	1572	+ 0,416

Giova mettere in evidenza come il metodo eterodina consenta una finezza di risoluzione frequenziale impossibile da ottenere, specie in regime transitorio, mediante l'analisi di Fourier: non per nulla il filtraggio eterodina, anche nella sua forma numerica, e` estesamente impiegato nei sistemi di telcomunicazione e di analisi spettrale che esigono le maggiori precisioni.

Le figure 3 e successive illustrano alcuni altri risultati ottenuti, e precisamente forniscono informazioni relative ai registri Principale 8' (figure 3) e Ottava 4' (figure 4) (questi due registri sono rappresentativi della famiglia dei Principali), Contrabbasso 16' + Ottava di contrabbasso 8' (rappresentativi della famiglia delle Viole, figure 5) e Trombone 8' (rappresentativo della famiglia delle Ance, figure 6).

In ciascun gruppo di figure la lettera a) individua il diagramma polare che mette in evidenza le variazioni del timbro @ e della complessita` C rilevabili tra i suoni a regime permanente di tutte le note esaminate per ciascun registro; la lettera b) individua, solo per una nota di ciascun registro, il diagramma polare che mette in evidenza le microvariazioni di timbro e complessita` durante il transitorio di attacco del suono; le lettere c) e d) indicano (solo per il Contrabbasso ed il Trombone e solo per la stessa nota di cui si e` illustrato il transitorio di attacco) i diagrammi degli scostamenti presentati dall'ampiezza e dalla frequenza della fondamentale rispetto ai valori nominali di regime: tali scostamenti sono stati ricavati con il metodo di filtraggio eterodina.

L'esame delle figure a) mette subito in evidenza che, tra le diverse note di uno stesso registro, esistono notevoli variazioni di timbro, mentre le variazioni di complessita` appaiono piu` contenute. Tuttavia le variazioni di timbro rimangono generalmente comprese nel settore caratteristico della famiglia di registri a cui il registro appartiene: tutte le note del Principale 8' presentano un timbro che cade nel settore dei Principali (fig.3a); tutte quelle del Contrabbasso 16' hanno un timbro che cade nel settore delle Viole (fig.5a); tutte quelle del Trombone 8' (fig. 6a) cadono nel settore delle Ance. Fa eccezione, invece, l'Ottava 4' (fig. 4a), le cui note presentano timbri che spaziano dal settore dei Flauti (nota n. 10 = FA#7), a quello dei Bordoni (nota n. 8 = FA#6), a quello delle Viole (nota n. 3 = DO4).

Per quanto riguarda le microvariazioni durante il transitorio di attacco le figure contraddistinte dalla lettera b) (si noti che Fig. 3b coincide con Fig. 1c) mettono in evidenza che, nei suoni delle canne labiali presi in considertazione (DO4 per il Principale 8' e l'Ottava 4', DO2 per il Contrabbasso 16'), la complessita` parte sempre da valori elevati e si assesta poi ai valori piu' bassi, tipici del comportamento a regime. Cio` e` in accordo con il fatto, ben noto, che le canne labiali inizialmente tendono ad emettere parziali di ordine piu` elevato, mentre solo a regime prevalgono la fondamentale e le parziali di ordine piu` basso. Anche il timbro, in transitorio, risente di questi fattori e presenta una tipica rotazione in senso antiorario, piu' marcata per il DO4 del principale 8', meno marcata per il DO4 dell'Ottava 4' e praticamente assente per il DO2 del Contrabbasso 16'.

Tutte le altre analisi effettuate, e qui non riportate per brevita`, tendono a confermare la legge della diminuzione della complessita` e della rotazione antioraria del timbro, durante il transitorio di attacco, propria delle canne labiali esaminate. Per le Ance, invece, si e` osservata una complessita` stabile e molto elevata ed un timbro pure assai stabile durante il transitorio di attacco. Nella fig. 4b), relativa al Trombone 8', si distinguono solo i punti n. 1 (inizio del suono) e n. 2 (20 ms dopo l'inizio); poi tutti gli altri punti si confondono in un ammasso compatto, che indica che la condizione di regime viene raggiunta dopo 40 ms dall'inizio ed e` molto stabile.

Le microvariazioni di ampiezza e frequenza delle fondamentali delle canne labiali, appartenenti alla famiglia dei Principali, sono apparse generalmente modeste; di solito crescono con il crescere del numero d'ordine delle parziali. E` interessante notare invece che le note esaminate del Contrabbasso 16' e del Trombone 8' presentano sensibili microvariazioni di ampiezza e frequenza (figure 5c, 5d, 6c, 6d), pur possedendo una notevole stabilita` del timbro durante il transitorio di attacco.

Passando a considerazioni di ordine piu` generale, tutti i registri hanno evidenziato microvariazioni caratteristiche per ciascuno di essi; in transitorio i registri ad ancia presentano minori variazioni rispetto ai registri labiali. Inoltre, se le fluttuazioni dell'ampiezza sono ridotte o praticamente assenti, anche le fluttuazioni della frequenza sono minori, benche` sempre presenti. Comparando tra loro le note analizzate per ciascun registro, sono pure apparse notevoli differenze nel timbro e nella complessita` (pur restando, in genere, nel settore del diagramma polare di pertinenza): cio` potrebbe attribuirsi a differente intonazione delle canne di uno stesso registro.

Quindi, per l'organo Callido di Gambarare, sembra si possa affermare che sensibili variazioni timbriche e di complessita` hanno luogo non solo passando da un registro ad un altro, ma anche, per uno stesso registro, al variare dell'altezza e, per una stessa nota, durante il transitorio di attacco.

Rimane una questione aperta se questa grande variabilita` non costituisca proprio una delle caratteristiche peculiari degli organi dell'epoca barocca, contribuendo a creare il fascino e la vivacita` tipici del loro suono.

L'applicazione dei metodi di misura descritti ad altri esemplari dell'arte organaria barocca potrebbe fornire una risposta al quesito, e si intende per l'appunto procedere su questa strada. Inoltre e` in corso uno studio per giungere ad una descrizione analitica unificata delle microvariazioni del tipo di quelle sinora riscontrate.

5 - CONCLUSIONI

Per concludere, sembra di poter affermare che i due metodi di studio delle microvariazioni dei suoni, sia a regime permanente, sia in transitorio, si integrano efficacemente consentendo di effettuare sia valutazioni di carattere complessivo, sia misurazioni analitiche di buona precisione, utili anche in vista di studiare metodi sempre piu` efficaci e naturali di sintesi numerica dei suoni. La loro applicazione allo studio delle caratteristiche foniche dell'organo Callido di Gambarare e` risultata molto soddisfacente e significativa, sia dal punto di vista della funzionalita` dei metodi di analisi impiegati, sia dal punto di vista dell'entita` ed originalita` dei risultati gia` ottenuti e che si ritiene di poter ulteriomente raggiungere.

Si desidera ringraziare il Parroco di Gambarare, Don Ralino Longhin, e l'ing. Mario Piccinelli per la preziosa collaborazione offerta durante la fase di registrazione dei suoni; si ringrazia pure la General Music S.p.A. per il supporto fornito al presente lavoro.

BIBLIOGRAFIA
[1] C.Chafe, "Pulsed Noise in Self-suistained Oscillations of Musical Instruments", CII2847-2/90-p.1157,1160-1990 IEEE.
[2] R. T. Schumacher and C. Chafe, "Characterisation of Aperiodicity in Nearly Periodic Signals", CH2847-2/90-p.1161,1164-1990 IEEE.
[3] T.Kobayashi and H. Sekine, "Statistical Properties of Fluctuation of Pitch Intervals and its Modeling for Natural Synthetic Speech", CH2847-2/90-p.321,324-1990 IEEE.
[4] G. B. Debiasi, M. Dal Sasso, "Metodo per la Valutazione Automatica dei Timbri di Organi a Canne", Atti di VIII C.I.M., Cagliari 1989.
[5] J. A. Moorer, "The Etherodyne Filter as a Tool for Analysis of Transient Waveforms", Stanford A. I. Lab., MEMO-AIM-208.
[6] M. D. Freedman, "A Method for Analysing Musical Tones", J.Audio Eng.Soc.vol.16 n.4,1968.
[7] J. S. Keeler, "The Attack Transient of Some Organ Pipes", IEEE AU 20 n.5-1972.
[8] M. Pullin, "Sistema di Analisi Numerica di Suoni Musicali", Tesi laurea Ingegneria elettronica A.A. 1983/84, Universita' di Padova.

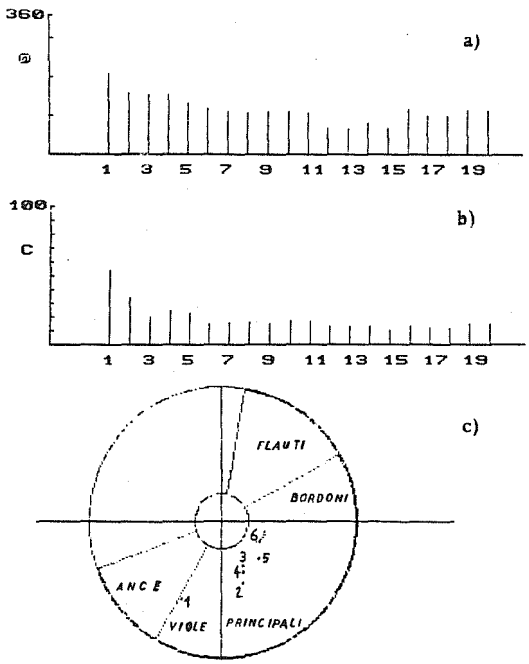


Fig. 1 -
a) Evoluzione temporale del timbro @;
b) Evoluzione temporale della complessita' C;
c) Diagramma polare.

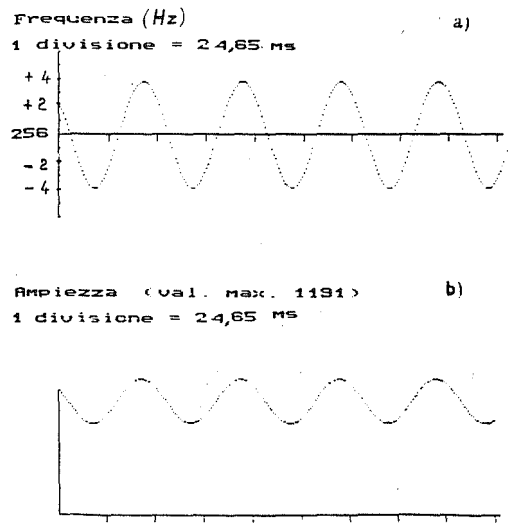


Fig. 2 -
a) Scostamenti della frequenza;
b) Scostamenti dell'ampiezza.

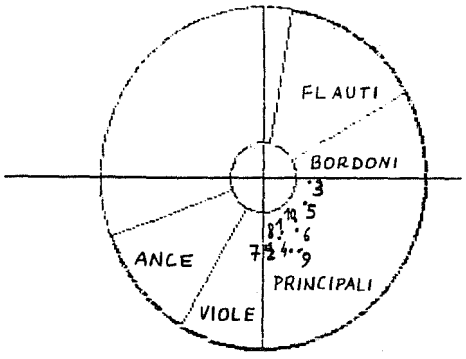


Fig. 3 a) - Analisi in regime stazionario;
Principale 8': 1=DO2,...,10=FA#6.

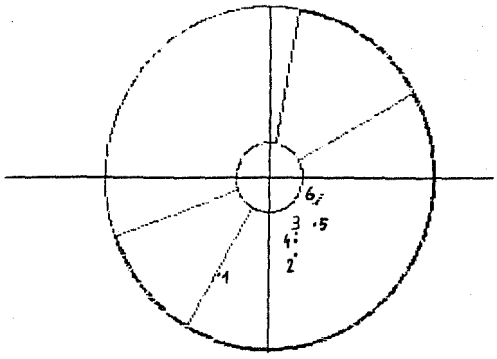


Fig.3 b) - Analisi della fase di attacco;
Principale 8': DO4.

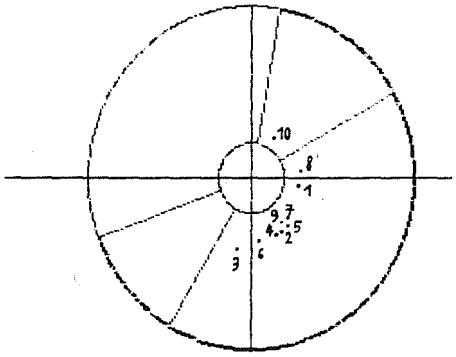


Fig. 4 a) - Analisi in regime stazionario;
Ottava 4': 1=DO3.....,10=FA#7.

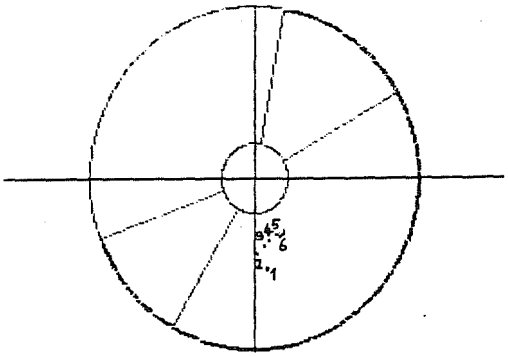


Fig. 4 b) - Analisi della fase di attacco;
Ottava 4': DO4.

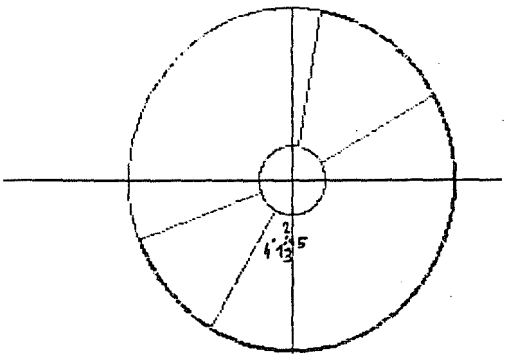


Fig. 5 a) - Analisi in regime stazionario;
Contrabbasso 16' + Ottava di Ctb. 8':
1=DO1,....,5=DO3.

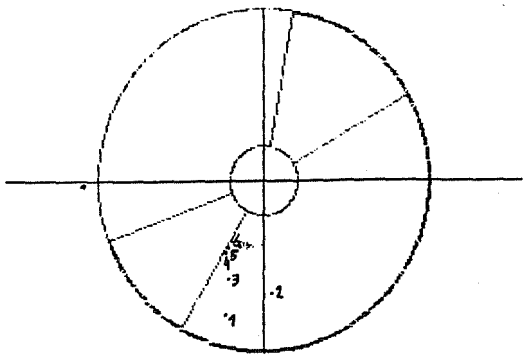


Fig. 5 b) - Analisi della fase di attacco;
Contrabbasso 16' + Ottava di Ctb. 8':
DO2.

Ampiezza (val. max. 5452)
Finestra temporale = 2.3751023751E+02 ms
1 divisione = 2.3751023751E+01 ms

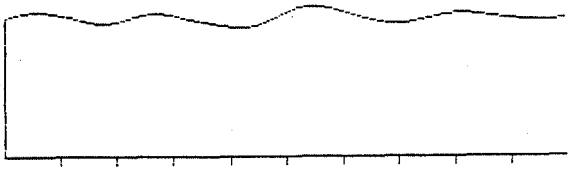


Fig. 5 c) - Scostamenti dell'ampiezza;
Contrabbasso 16' + Ottava di Ctb. 8':
DO2.

Frequenza
Finestra temporale = 2.3751023751E+02 ms
1 divisione = 2.3751023751E+01 ms

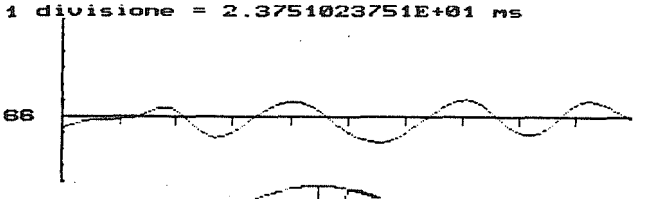


Fig 5 d) - Scostamenti della frequenza;
Contrabbasso 16' + Ottava di Ctb. 8':
DO2.

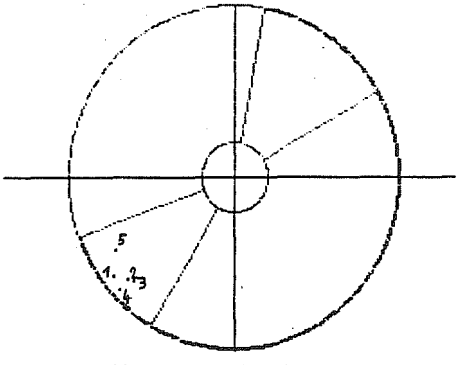


Fig. 6 a) - Analisi in regime stazionario;
Trombone 8': 1=DO1,...,5=DO3.

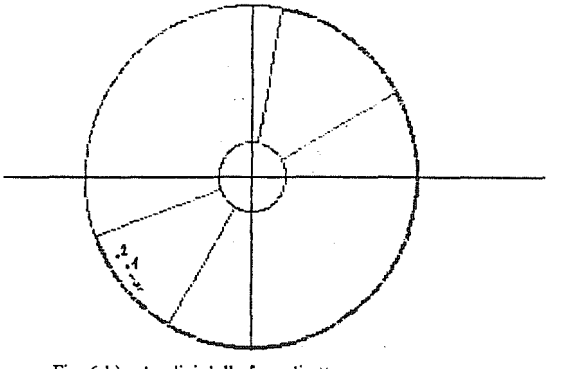
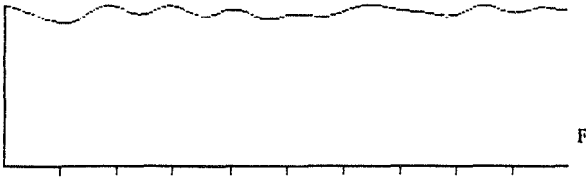


Fig. 6 b) - Analisi della fase di attacco;
Trombone 8': FA#2.

Ampiezza (val. max. 4306)
Finestra temporale = 2.3751023751E+02 ms
1 divisione = 2.3751023751E+01 ms



Fig, 6 c) - Scostamenti dell'ampiezza;
Trombone 8': FA#2.

Frequenza
Finestra temporale = 2.3751023751E+02 ms
1 divisione = 2.3751023751E+01 ms

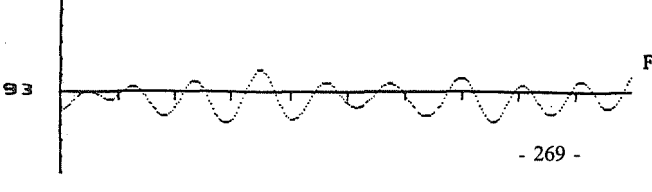


Fig. 6 d) - Scostamenti della frequenza;
Trombone 8': FA#2.

APPENDICE

Il filtro eterodina consente di estrarre le componenti spettrali presenti in un suono, una alla volta, in modo da poterne seguire l'evoluzione nel tempo dell'ampiezza e della frequenza.

FONDAMENTI TEORICI

Si consideri un segnale campionato:

$$s(k) = s(t), \quad \text{per } t = kT; \tag{a.0}$$

$f_c = 1/T$ e' la frequenza di campionamento.
Le formule per il calcolo dell'ampiezza e della frequenza delle singole armoniche sono le seguenti:

$$a_n(k) = \sum_{i=k}^{k+N-1} s(i)*\cos(in\omega_o T) \tag{a.1}$$

$$b_n(k) = \sum_{i=k}^{k+N-1} s(i)*\sin(in\omega_o T) \tag{a.2}$$

$$\hat{A}_n(k) = \sqrt{a_n(k)^2 + b_n(k)^2} \tag{a.3}$$

$$\hat{\phi}_n(k) = \arctan\left[\frac{b_n(k)}{a_n(k)}\right] \tag{a.4}$$

$$\hat{F}_n(k) = \frac{1}{2\pi} * \left[n\omega_o + \frac{d\hat{\phi}_n(k)}{dt}\right]. \tag{a.5}$$

In tali formule: $n\omega_o$ e' la frequenza angolare (o pulsazione) di analisi (in rad/s); n e' l'ordine dell'armonica che si

intende analizzare; s(i) e' il valore del segnale d'ingresso all'istante iT; N e' il numero intero piu' prossimo al numero di campioni per periodo del segnale d'ingresso; $a_n(k)$, $b_n(k)$, $\hat{A}_n(k)$, $\hat{\phi}_n(k)$, $\hat{F}_n(k)$ sono, rispettivamente, la parte reale, la parte immaginaria, l'ampiezza, la fase e la frequenza istantanea dell'armonica di ordine n, all'istante kT.

Calcolando la sommatoria, in $a_n(k)$ e $b_n(k)$, su N campioni, si pongono degli zeri di trasmissione in corrispondenza alle frequenze delle armoniche che non sono sotto analisi; il che significa che i contributi di tali armoniche nella sommatoria risultano nulli.

Quindi, $a_n(k)$ e $b_n(k)$ rappresentano, effettivamente, la parte reale e la parte immaginaria della sola armonica in esame.

Dato che, nei calcoli, il periodo della fondamentale risulta determinato da N, si deve porre:

$$\omega_o = \frac{2\pi}{NT} \quad , \qquad (a.6)$$

in modo che il periodo dell'armonica e la frequenza di analisi risultino esattamente correlati.

Pero', cosi' facendo, si analizza il segnale ad una frequenza che potrebbe non coincidere esattamente con quella effettiva del suono.

Il simbolo " ^ " (in $\hat{A}$, $\hat{\phi}$, $\hat{F}$) indica, appunto, che queste grandezze costituiscono soltanto delle *stime* dei valori effettivi.

Infatti, se la frequenza di analisi si discosta notevolmente dall'altezza effettiva del suono, gli zeri di trasmissione non vengono posti esattamente in corrispondenza alle frequenze delle armoniche che dovrebbero venire cancellate.

Ne consegue che i risultati dell'analisi non sono relativi ad una singola armonica, bensi' ad una sorta di combinazione non lineare delle varie armoniche.

Tuttavia, se N e' abbastanza grande, le grandezze stimate si avvicinano molto a quelle effettive e l'errore commesso si puo' ritenere trascurabile. Percio', occorre scegliere f_c in modo che N non scenda a valori troppo bassi (indicativamente, non inferiori a 50).

DESCRIZIONE DEL PROGRAMMA

Inizialmente il programma richiede alcune informazioni: il nome del file in cui sono memorizzati i campioni del segnale d'ingresso; quali nomi si desidera assegnare ai file che il programma fornisce in uscita; la frequenza di campionamento f_c; una stima della frequenza della fondamentale f_s; l'ordine dell'armonica che si intende analizzare n.

Si calcola N come il numero intero che meglio approssima f_c/f_s, cioe' il numero di campioni per periodo.

Noto N, si calcola il valore della frequenza di analisi: $f_o = f_c/N$, e, quindi: $\omega_o = 2\pi f_o$.

Per risparmiare tempo di calcolo, si memorizzano in 2 vettori i valori del coseno e del seno, che ,successivamente, vengono utilizzati per il calcolo di $a_n(k)$ e $b_n(k)$.

Anche nei calcoli relativi ad armoniche di ordine superiore, si fa sempre riferimento ad un periodo della fondamentale. Infatti, per le armoniche di ordine superiore, all'aumentare della frequenza, diminuisce il numero di campioni per periodo ed i calcoli perdono di significato. Invece, considerando un periodo della fondamentale, i calcoli si eseguono su un numero di campioni multiplo intero del numero di campioni per periodo dell'armonica in esame e, quindi, sufficientemente elevato.

I campioni relativi al segnale in esame vengono letti dal file di ingresso e memorizzati in uno o piu' vettori.

I campioni del segnale d'ingresso vengono opportunamente moltiplicati per i valori, memorizzati in precedenza, del

coseno e del seno. Poi, i prodotti vengono mediati su di un periodo della fondamentale.

Per risparmiare tempo di calcolo, le sommatorie in $a_n(k)$ e $b_n(k)$ vengono calcolate non per ogni valore di k, bensi' soltanto per k=1, cioe' per il primo periodo; poi, si sottrae il termine relativo all'istante i=k=1 e si somma quello relativo all'istante i=k+N=N+1: in questo modo si ottengono $a_n(2)$ e $b_n(2)$; iterando il procedimento per k=2,3,..., si ottengono tutti gli $a_n(k)$ ed i $b_n(k)$ e, per ogni k, si devono eseguire soltanto una sottrazione ed una addizione.

Cosi', si memorizzano in due vettori i valori della parte reale e della parte immaginaria dell'armonica in esame.

Per ottimizzare le prestazioni del filtro, si calcolano le medie per periodo di tali valori per tre volte; infatti, cio' accentua l'effetto dello zero di trasmissione in corrispondenza alle armoniche che non sono sotto analisi. Dato che N e' soltanto un'approssimazione del numero di campioni per periodo, le medie vengono calcolate su N-1, N, N+1 campioni per periodo. In questo modo, si cerca di ridurre gli errori dovuti alle approssimazioni. Per il calcolo delle medie, si applica il procedimento appena illustrato.

Cosi', si memorizzano in due vettori i valori di $\bar{a}_n(k)$ e $\bar{b}_n(k)$, cioe' parte reale e parte immaginaria mediate, dell'armonica di ordine n.

L'ampiezza dell'armonica di ordine n si calcola mediante la formula:

$$\hat{A}_n(k) = \sqrt{\bar{a}_n(k)^2 + \bar{b}_n(k)^2} \quad . \qquad (a.7)$$

Calcolando la fase mediante la formula:

$$\hat{\phi}_n(k) = \arctan\left[\frac{\bar{b}_n(k)}{\bar{a}_n(k)}\right] , \qquad (a.8)$$

si verificano dei salti ad intervalli di π e, di conseguenza, problemi nel calcolo della derivata.

Per ottenere una funzione continua si ricorre ad un algoritmo che calcola, anziche' l'angolo di fase punto per punto, la differenza rispetto all'angolo di fase nel punto precedente. Le formule sono le seguenti:

$$\cos(\phi_{n,k}) = \frac{\bar{a}_n(k)}{\sqrt{\bar{a}_n(k)^2 + \bar{b}_n(k)^2}} \qquad (a.9)$$

$$\sin(\phi_{n,k}) = \frac{\bar{b}_n(k)}{\sqrt{\bar{a}_n(k)^2 + \bar{b}_n(k)^2}} \qquad (a.10)$$

$$\sin(\Delta\vartheta) = \sin(\phi_{n,k}) * \cos(\phi_{n,k-1}) - \cos(\phi_{n,k}) * \sin(\phi_{n,k-1}) \qquad (a.11)$$

$$\cos(\Delta\vartheta) = \cos(\phi_{n,k}) * \cos(\phi_{n,k-1}) + \sin(\phi_{n,k}) * \sin(\phi_{n,k-1}) \qquad (a.12)$$

$$\Delta\vartheta = \arctan\left[\frac{\sin(\Delta\vartheta)}{\cos(\Delta\vartheta)}\right] \qquad (a.13)$$

$$\vartheta_{n,k} = \vartheta_{n,k-1} + \Delta\vartheta \quad , \qquad (a.14)$$

in cui: $\phi_{n,k}$ e' l'angolo di fase dell'armonica di ordine n all'istante kT; $\vartheta_{n,k}$ e' l'angolo di fase dell' armonica di ordine n all'istante kT, posta la condizione iniziale: $\vartheta_{n,0} = 0$; $\Delta\vartheta$ e' la variazione dell'angolo di fase dell'armonica di ordine n tra l'istante (k-1)T e l'istante kT.

Ora, si puo' calcolare la derivata della fase come rapporto incrementale; quindi, la frequenza istantanea e' data da:

$$\hat{F}_n(k) = \frac{1}{2\pi} * \left\{ \omega_o + \frac{\hat{\vartheta}_n(k) - \hat{\vartheta}_n(k-1)}{T} \right\} \quad . \quad (a.15)$$

I valori dell'ampiezza, della fase e della frequenza, uno ogni dieci, vengono memorizzati, sotto forma di tabella, in un file che, successivamente, si puo' visualizzare.
Inoltre, i valori dell'ampiezza e della frequenza, opportunamente normalizzati, vengono memorizzati in due file di numeri interi, che servono per tracciarne direttamente i grafici. Tramite opportuni programmi e' possibile imporre la risoluzione desiderata. Inoltre, e' pure possibile scegliere un fattore di scalamento, per l'asse delle ordinate, nella visualizzazione della frequenza.

Physical Models of Woodwind Instruments in a Representation by means of Wave Digital Filters

F. D'Agostino†
I. Ortosecco‡
†Cap Gemini SESA Italia – Roma
‡Dip. Scienze Fisiche – Università "Federico II", Napoli
E-mail:acel@napoli.infn.it

June 1991

Keywords: Physical Models,Woodwind Instruments ,Transmission Line , Wave Digital Filters ,Scattering Matrix.

Abstract

The increasing interest in developing physical models of musical instruments is fed both by the need for studying physical phaenomena, from which sound production of these acoustical systems rises, and by the great growth in computer and digital audio devices technology; the latter allowed the development of simulations of the sound production systems dynamics in implementations on computers.

The physical model of a musical instrument must simulate the motion laws of the sound vibrating generator; hence, it must take into account both the linear response of the vibrating medium and the nonlinear effects produced by the control device of vibrations.

In a Digital Signal Processing (DSP) system (the set of data and programs in a computer), the linear response of the vibrating medium may be produced by a digital filter, while the control device is simulated by the numeric solution of the nonlinear equations describing it.

In this paper we show an implementation of woodwind instruments physical models in which, starting from the Transmission Line Model (TLM) of the air column, the linear behavior component is represented by a Wave Digital Filter (WDF), giving rise to a powerful, although formally simple, representation in a DSP system. While outlining the theoretical framework giving rise to our formulation, we show the relations between WDF structures and physical parameters describing the main features of the air column of the woodwind instrument.

1 Physical models of woodwind instruments.

In woodwind instruments an air column inside the pipe is excited in its vibrational modes by the motion of a nonlinear flow–control device (the *reed*), which can be viewed as a pressure–controlled valve [Benade 76]: the reed's motion, supplied by the player's puff, quickly brings the air column to its steady–state oscillations, producing and mantaining the so–called "self–sustained" oscillations [Benade Kouz. 88].

The air column has a linear behavior: its resonances depend upon its physical dimensions and the sound speed in the air.

The behavior of the reed, which motion is pressure–controlled by the air column response, is nonlinear; the state of the acoustical system may be described by the pressure $q(t)$ just into the instrument's embouchure and the volume velocity $f(t)$, related by the *Backus' law* [Backus 63]:

$$f = F(q) \tag{1}$$

where F is a nonlinear function of q.

The air column is described in a straightforward way by the wave formulation. The pressure $q(t)$ is the sum of an incoming and an outcoming wave (with respect to the embouchure):

$$q(x,t) = q_i(t + x/c) + q_o(t - x/c) \tag{2}$$

where c is the sound speed in the air and x is the distance from the embouchure along the column.

The volume velocity $f(t)$ is given by:

$$f(x,t) = [q_o(t - x/c) + q_i(t + x/c)]/Z \tag{3}$$

where Z is the characteristic impedance of the air column.

By making use of the linearity of the air column, from eqs.2 and 3 we obtain, for $x = 0$, i.e. in the embouchure [McIntyre *et al.* 83]:

$$q(t) = q_h(t) + Zf(t) \tag{4}$$

where:

$$q_h(t) = r(t) * [q(t) + Zf(t)] = \int_0^\infty r(t')[q(t - t') + Zf(t - t')]dt' \tag{5}$$

is the *pressure history function* and $r(t)$ is the *reflection function*, defined as the impulse response of the air column; the reflection function is defined only for $t \geq 0$ and has the following property:

$$\int_0^\infty r(t)dt = -1 \tag{6}$$

The overall behavior of the air column may be splitted in a lossless pipe response and in a low–pass effect due to viscous and thermal dispersions along pipe walls at tonehole placements.

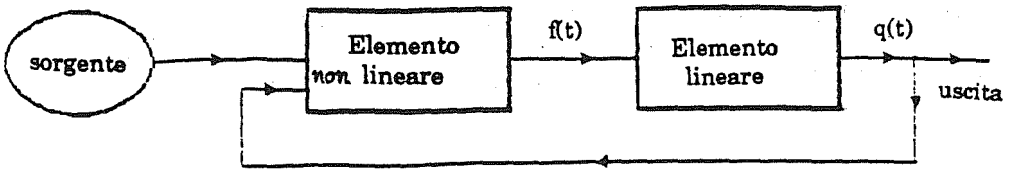


Figure 1: Block diagram of a physical model of a woodwind instrument.

The latter is well described in terms of the *open tonehole cutoff frequency*, related to physical dimensions of the toneholes and to their spacing. For the clarinet this cutoff frequency is:

$$f_c \approx 0.11 \frac{b}{a} c \left(\frac{1}{s t_e}\right)^{1/2} \tag{7}$$

where a is the bore radius, b the tonehole radius, t_e the tonehole equivalent height, $2s$ the interhole spacing.

The block diagram of the physical model of a woodwind instrument is in Fig.1, where the feedback loop represents the pressure input, coming from the air column, to the flow control device.

The physical model must compute, at any time t, the solution of the system of eqs.1, 4 and 5.

In our implementation the physical model is a Digital Signal Processing (DSP) system [Opp. Scha. 75, Rab. Gold 75], described by *discrete-time digital signals* (i.e. sequences of numbers) $x(n)$, related to the continuous-time signals $x_c(t)$by:

$$x(n) = \mathcal{F}[x_c(nT)]$$

where T is the *sampling period* and $\mathcal{F}$ is the (nonlinear) operator giving the AD-convertion and the floating-point representation of the analog signal.

The convolution, eq. 5, becomes then the discrete-time convolution:

$$q_h(n) = \sum_{m=0}^{\infty} r(m) x(n-m) \tag{8}$$

where we let:

$$x(n) = q(n) + Zf(n)$$

From the DSP point of view, eq. 8 means that the linear component response is the output of a digital filter, having impulse response $r(n)$ and input $x(n)$. Moreover, eq. 8 shows that this digital filter must be *causal* and *of infinite duration*, i.e. an IIR filter; from the normalization condition, the discrete-time version of eq. 6:

$$\sum_{n=0}^{\infty} r(n) = -1 \tag{9}$$

the filter must also be *stable*.

In the next section we'll discuss a continuous–time representation of the air column that yields our Wave Digital Filter (WDF) representation by means of a formal analogy.

2 Transmission line model of the air column.

A woodwind instrument is composed by a sequence of pipes and cavities: by means of electroacoustical analogy [Hunter 57], the resulting acoustical network may be represented by an electrical network.

The air column is a sound waves propagation medium: as such, it may be described in terms of incident and reflected sound waves. The electrical network described in terms of (voltage and current) incident and reflected waves is the *transmission line* [Mill. Taub 65]: hence, we are able to derive a Transmission Line Model (TLM) of the air column, taking into account not only the delay effects, which are one of the most interesting features of trasmission lines, but also the dispersion, due to thermal and frictional dissipation along the pipe and across the instrument's toneholes, and radiation at instrument's flaring bell.

The details about tonehole effects are not in our concern of deriving a formal DSP representation of the air column in our model: they may be found e.g. in [Keefe 82a, Keefe 82b], while in [Keefe 90] a detailed TLM of woodwind air columns is described, together with an algorithm to obtain the reflection function $r(t)$ from the TLM: hence, the discrete–time sequence $r(n)$ is got by digital sampling of $r(t)$.

As we'll see in the next section, we'll use the TLM representation as starting point of our algorithm. The final result – the WDF representation – is achieved *by domain transformations, not by digital sampling of a continuous–time function.*

A powerful description of the TLM is achieved by using the *transfer matrix* representation of a trasmission line. It states the voltage and current wave relations between the input and the output end of the line: in the state representation of a woodwind instrument, it corresponds to get *pressure and volume velocity wave* relations between two points along the air column.

So, letting q, f and $\mathbf{K}$ pressure, volume velocity and the 2×2 trasfer matrix of the line of length x respectively, we have:

$$\begin{pmatrix} q_{out} \\ f_{out} \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} q_{in} \\ f_{in} \end{pmatrix} = \mathbf{K} \begin{pmatrix} q_{in} \\ f_{in} \end{pmatrix} \tag{10}$$

where:

$$\begin{aligned} A &= \cosh(\gamma x); \qquad B = Z_c \sinh(\gamma x) \\ C &= Z_c^{-1} \sinh(\gamma x); \quad D = \cosh(\gamma x) \end{aligned} \tag{11}$$

γ being the propagation wave number and Z_c the characteristic impedance of the line.

Eqs. 10 and 11 are a full description of an acoustical transmission line, which represents a simple as detailed description of the linear behavior of woodwind air columns.

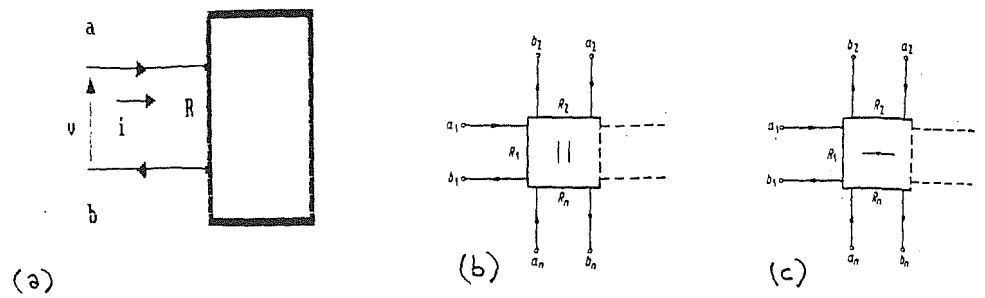


Figure 2: (a) Block diagram of a WDF port. (b) WDF diagram of the parallel adaptor. (c) WDF diagram of the series adaptor.

3 WDF representation of the air column.

In this section we show our derivation of a DSP representation of the linear behavior component from the TLM of the air column. The generation of *formal* DSP description is made possible by introducing a structure of digital filters, the WDFs, which share a wave formulation with transmission lines.

Before entering the details of our formulation, a brief overview on WDFs is needed.

3.1 WDF background concepts.

WDFs [Fettweis 71, Fettweis 86] are digital filter structures based on an analogy with classical networks [Belevitch 68]. The correspondance between analog and digital domain is set by means of domain transformations; from the analog s–plane the WDF *reference domain* ψ is defined as:

$$\psi = \tanh(sT/2) \tag{12}$$

where T is the sampling period of the digital structure.

Letting $z = e^{sT}$ the digital z–plane, the relation between reference and digital domain holds via the bilinear transform:

$$\psi = \frac{z-1}{z+1} \tag{13}$$

The building block of a WDF is called *port*, which is plotted if Fig.2(a): it has an input and an output end, each one described by a *voltage wave quantity*. $a(t)$ and $b(t)$ are called instantaneous *incident* and *reflected* wave quantities respectively; they are related to port voltage $v(t)$, current $i(t)$ and resistance R by the *wave equations*:

$$a(t) = v(t) + Ri(t) \quad b(t) = v(t) - Ri(t) \tag{14}$$

WDFs are described by the number of ports available from the external. The simplest WDF is thus the one–port building block: by means of domain transformations, eqs. 12 and 13, one gets the digital realization of a given classical one–port having a single circuit element.

Connections (i.e., formally, Kirchhoff's laws) are given in the WDF domain by structures called *adaptors* [Fett. Meer. 75], digital networks whose wave equations are obtained equating port voltages and currents in parallel (Fig.2(b)) and series (Fig.2(c)) adaptors respectively.

Given the structure of a classical network, it is straightforward to obtain the WDF derived by connections of building blocks. WDFs show very useful properties in the digital domain [Fettweis 72]: stability, low passband sensitivity to changes in multipliers, good dynamic range.

3.2 The two–port WDF.

Two–port WDFs are easily described in the reference ψ–plane in terms of the *scattering equations*, expressed in matrix form:

$$\mathbf{b} = \mathbf{S}\mathbf{a}$$

where

$$\mathbf{S} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}$$

is called the *scattering matrix* and

$$\mathbf{a} = \begin{pmatrix} A_1 & A_2 \end{pmatrix}^T \quad \mathbf{b} = \begin{pmatrix} B_1 & B_2 \end{pmatrix}^T$$

are the incident and reflected *wave vectors* respectively.

The z–plane representation, obtained by the bilinear transform of eq. 13, yields the time–domain implementation of the two–port WDF.

3.3 The WDF representation of woodwind air columns.

Our TLM of woodwind air columns represent an air column as a lossless acoustical transmission line of length L and delay time

$$t_d = L/c \tag{15}$$

c being the sound speed in the air, terminated by a radiation impedance [D'Agostino 91]; t_d equates the round-trip time of the sound wave into the pipe.

Let us show that the TLM formulation leads to a digital representation by means of a two–port WDF.

The transfer matrix of the lossless part (i.e. with $Z_c = R_0$, a pure resistance) of the transmission line is

$$\mathbf{K} = \begin{pmatrix} \cosh(\gamma L) & R_0 \sinh(\gamma L) \\ R_0^{-1} \sinh(\gamma L) & \cosh(\gamma L) \end{pmatrix} \tag{16}$$

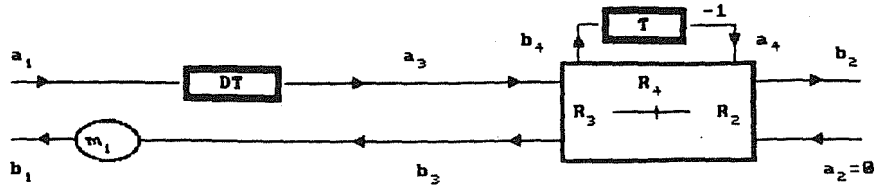


Figure 3: WDF model of the air column of a woodwind instrument.

Consider a lossless transmission line of length $x = cT/2$, where T is the sampling period of a DSP system. Setting $\gamma = s/c$ and using eq.12, we obtain from eq. 16:

$$\mathbf{K}(\psi) = \frac{1}{\sqrt{1-\psi^2}} \begin{pmatrix} 1 & R_0\psi \\ R_0^{-1}\psi & 1 \end{pmatrix} \tag{17}$$

Next we express the pressure (voltage) and volume velocity (current) waves at input and output ends in terms of voltage incident and reflected wave quantities of a symmetrical (i.e., with equal port resistances) two–port WDF:

$$q_i = \frac{a_i + b_i}{R} f_i = \frac{a_i - b_i}{R} \tag{18}$$

$i = 1, 2$, where '1' corresponds to 'out' and '2' to 'in' in eq.10.

Introducing eqs. 17 and 18 into eq. 10, we obtain the scattering equations of the two–port WDF representing an acoustical lossless transmission line of length $x = cT/2$; the scattering matrix is:

$$\mathbf{S_L} = \begin{pmatrix} 0 & \sqrt{\frac{1+\psi}{1-\psi}} \\ \sqrt{\frac{1+\psi}{1-\psi}} & 0 \end{pmatrix}$$

whose WDF realization is a one sample delay from a_1 to b_2.

The WDF model of the air column is shown in Fig.3. The right part implements the radiation impedance, while the D–samples delay is obtained by concatenation of D two–port WDFs, where:

$$D = \frac{2L}{cT} \tag{19}$$

L being the air column length. The multiplier m_1 is valued by the normalization condition, eq. 9.

The tonehole low–pass effect may be taken into account apart from the air column WDF model by means of a low–pass filter whose cutoff frequency equals the open tonehole cutoff frequency, eq. 7.

The choice of a low–pass WDF provides omogeneity to our model, which is thus *entirely* designed by means of WDF structures; lattice WDFs (LWDFs) [Fettweis *et al.* 74] show properties, such as *branching*, very low passband sensitivity and modularity in implementation, but above all they allow a scattering matrix description very straightforward both in design and in implementation phases.

The design of a LWDF is carried out by solving a problem of polynomial approximation in the ψ–plane; coefficients of polynomials are evaluated by a design procedure, which is a Pascal implementation of the algorithm for low–pass LWDFs given in [Gazsi 85].

4 Applications of the WDF representation.

In this section we briefly show the application of our WDF model in woodwind instruments simulations, outlining the correspondance between physical parameters and WDF coefficients.

We implemented a clarinet model with typical physical parameters [Benade Kouz. 88]; Backus' law, for the clarinet, is [Keefe Park 90]:

$$f(n) = A\{|p(n)|[H - D^{-1}p(n)]^2\}^{2/3}\epsilon(p)\beta(p) \tag{20}$$

where A is a constant, $p = P - q$, P being the player's (external) pressure, H is reed's greatest displacement, D reed's stiffness and $\beta(p)$ the treshold function (zero for p less than the critical pressure).

The air column is represented by the WDF of Fig.3, while the low–pass effect is simulated by the elliptic, ninth order LWDF shown in Fig.4. Its cutoff frequency, computable from eq.7, is $1500Hz$.

In Fig.5 we show the output of the model and its spectrum. The clarinet–like spectrum has only the first and the third armonic frequencies, as we expect from the theory and experiments on real clarinets.

The length of the air column parameterizes the delay of the WDF air column model, cfr. eq. 19. The change of this parameter corresponds to the change of fingering to obtain different musical tones from different air column lengths.

The change of cutoff frequency yields a change in output sound thymber: it corresponds to modify some physical parameters (the ones involved in eq.7) of the simulated instrument. Fig. 6 shows output waveform and spectrum of the clarinet model with LWDF cutoff frequency at $3000Hz$.

The external (player supplied) pressure has a treshold behavior, as it has a lower bound which instrument doesn't play beyond (the reed can't start its motion). In physical model, this treshold effect is well observable and is another interesting feature that gives our WDF model great physical consistency. It's out of our concern to show results of this further parameterization in this paper: we were interested in giving some details about the new point of view provided by our formulation.

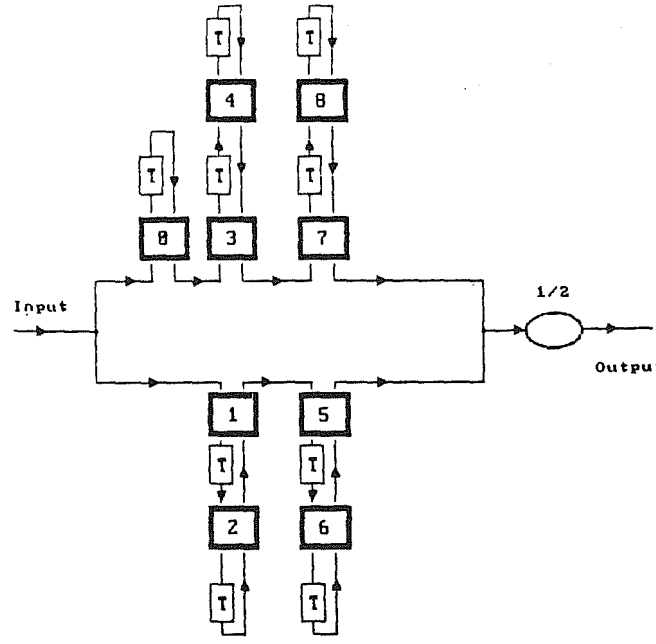


Figure 4: Low–pass elliptic, ninth order LWDF having $F_c \approx 1500Hz$.

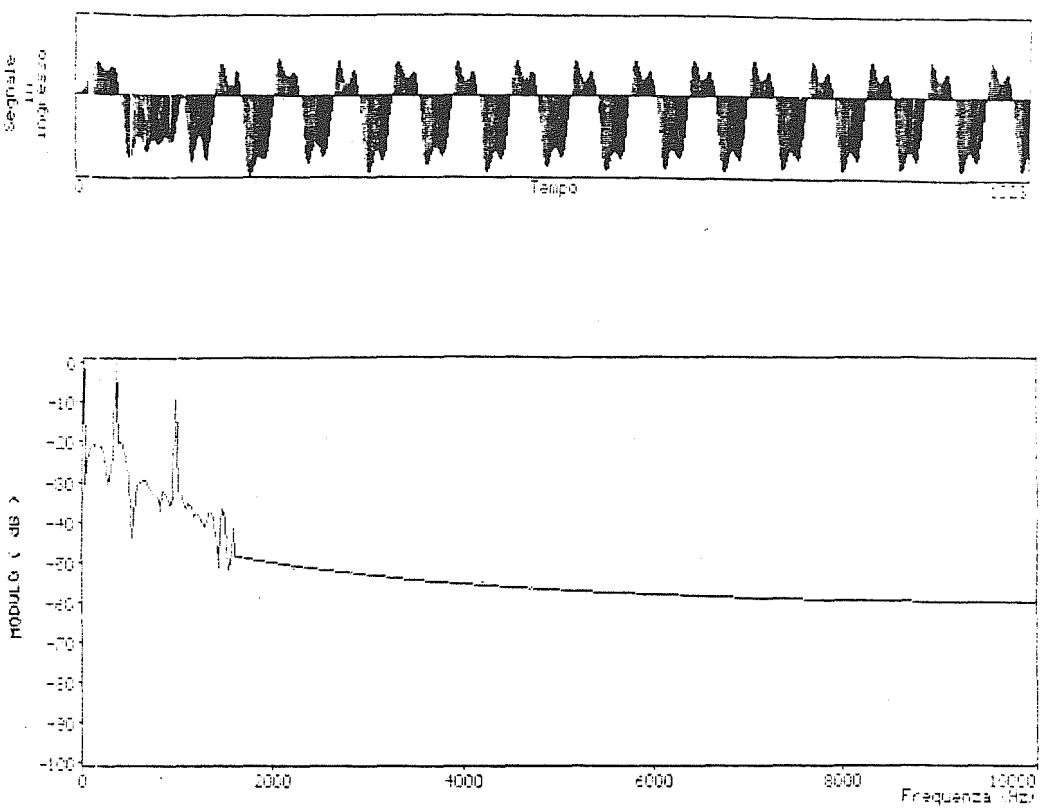


Figure 5: Output waveform and spectrum of a clarinet model (low–pass with $F_c \approx 1500Hz$).

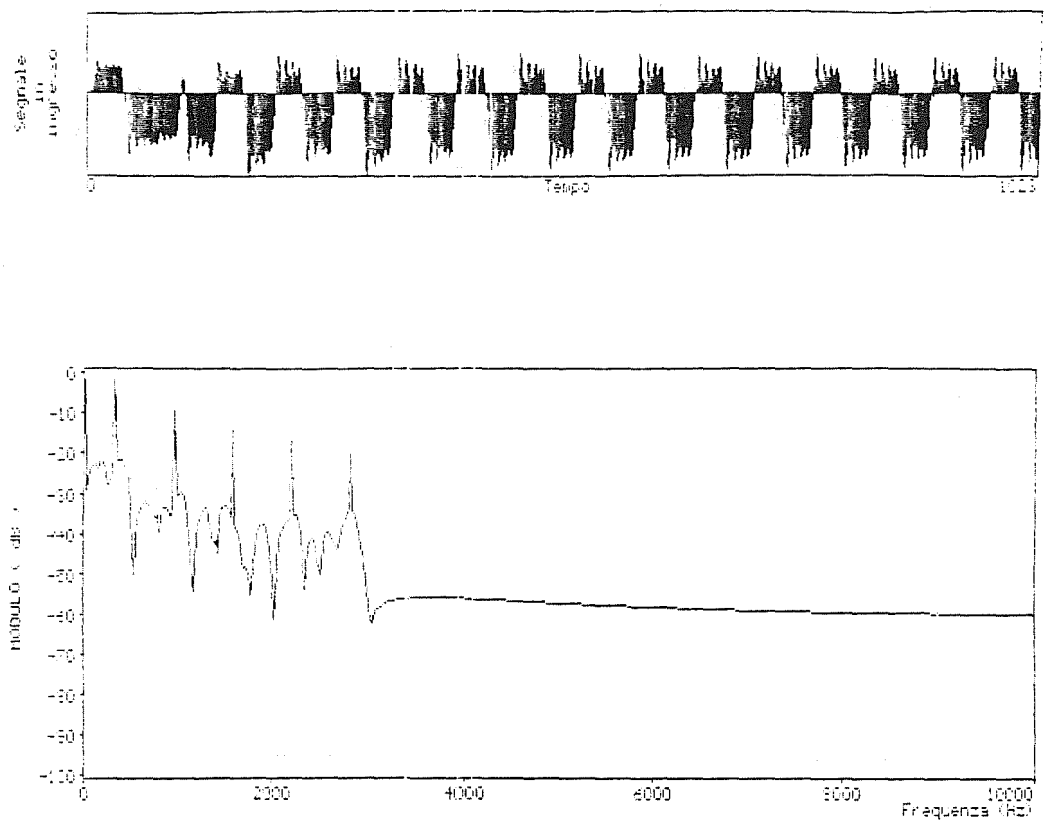


Figure 6: Output waveform and spectrum of a clarinet model (low–pass with $F_c \approx 3000Hz$).

5 Concluding remarks.

In this paper we have outlined a new formulation of physical models of woodwind instruments based on an implementation of the linear behavior component by means of WDFs.

Let us highlight the greater features of this formulation:

- The WDF model of the air column has a formally simple and smart description by means of scattering matrices.
- *Our WDF model is entirely parameterized by physical parameters of the woodwind instrument*: this gives an high degree of abstraction, as physical properties are related to mathematical properties of the scattering matrices.
- The formal description allows further developments, in terms of accuracy and efficiency, of physical models not only of woodwind instrument, but of other musical instruments too.
- The application of this representation to the realization of real time physical models should give great contributions to the future of computer music and of the monitoring on design and realization of musical instruments.

REFERENCES

A. H. Benade, **Fundamentals of musical acoustics.** Oxford University Press, 1976.

A. H. Benade, S. N. Kouzoupis, "The clarinet spectrum: theory and experiment", **Journ. Acoust. Soc. Am.**, vol.83, pp.292–304, 1988.

J. Backus, "Small-vibration theory of the clarinet", **Journ. Acoust. Soc. Am.**, vol.35, pp.305–313, 1963.

M. E. McIntyre, R. T. Schumacher, J. Woodhouse, "On the oscillations of musical instruments", **Journ. Acoust. Soc. Am.**, vol.74, pp.1325–1345, 1983.

A. V. Oppenheim, R. W. Schafer, *Digital Signal Processing.* Prentice–Hall, 1975.

L. R. Rabiner, B. Gold, **Theory and applications of Digital Signal Processing.** Prentice–Hall, 1975.

J. L. Hunter, **Acoustics.** Prentice–Hall, 1957.

J. Millman, H. Taub, **Pulse, digital and switching waveforms** McGraw–Hill, 1965.

D. H. Keefe, "Theory of the single woodwind tone hole", **Journ. Acoust. Soc. Am.**, vol.72, pp.676–687, 1982.

D. H. Keefe, "Experiments on the single woodwind tone hole", **Journ. Acoust. Soc. Am.**, vol.72, pp.688–699, 1982.

D. H. Keefe, "Woodwind air column models", **Journ. Acoust. Soc. Am.**, vol.88, pp.35–51, 1990.

A. Fettweis, "Digital filter structures related to classical filter networks", **Archiv Elektr. und Uebertr.** , vol.25, pp.79–89, 1971.

A. Fettweis, "Wave digital filters: theory and practice", **Proc. IEEE** , vol.74, pp.270–327, 1986.

V. Belevitch, **Classical network theory.** Holden Day, 1968.

A. Fettweis, K. Meerkötter, "On adaptors for wave digital filters", **IEEE Trans. on ASSP** , vol.23, pp.516–525, 1975.

A. Fettweis, "Pseudopassivity, sensitivity and stability of wave digital filters", **IEEE Trans. on Circuit Theory** , vol.19, pp.668–673, 1972.

F. D'Agostino, **Modelli fisici degli strumenti a fiato e della voce in una implementazione mediante Wave Digital Filters.** Doctoral Thesis, Physics Dept., University "Federico II", Napoli, 1991.

A. Fettweis, H. Levin, A. Sedlmeyer, "Wave digital lattice filters",

L. Gazsi, "Explicit formulas for lattice wave digital filters", **IEEE Trans. on Circuits and Systems** , vol.32, pp.68–88, 1985.

D. H. Keefe, M. Park, "Tutorial on physical models of wind instruments: II, Time Domain Simulations",**Technical Report no. 9002 of Systematic Musicology Program**, University of Washington, Seattle, 1990.

ANALYSIS/SYNTHESIS SYSTEM BASED ON A PERCEPTIVE MODEL

A.Piccialli, P.Basile, M.Sala, P.Simone, S.Vergara

Dipartimento di Scienze Fisiche Universita' di Napoli

ABSTRACT

Every analysis-synthesis system has an underlying model, generally this model conforms to certain physical properties determined by the particular system being modeled. In this paper a different point of view is introduced, using a filter-bank modeled after band relationship of human auditory perception. Recostrucion of signals is assured by the use of opportune filters. Short time analysis and synthesis systems are useful in audio signal processing and describe how a signal in the frequency domain evolves over time.

The limitation of a useful representation of audio-band signals are well known: we are referring particularly to vocalic sounds and some musical instruments sounds. These representations imply a model of the physical mechanism of sound production, i.e. the quasi-stationary representation for the speech waveform starts with a model for the production of speech subject to slow temporal changes in the vocal-tract geometry. Short-time analysis and synthesis are referred only to a class of processing signals slowly time-varying.

In this paper the authors propose an new analysis-synthesis system based rather than on the model of production of sounds, on a model conform to auditory sensitivity. Such a description of the musical and voice signals are very efficient in certain applications for recognizing or modifing parameters of interest.

Q constant filter-bank, in wich each bandpass filter has a band width proportional to its center frequency, has been considered by some authors; related problem are the recostruction of signals and the efficiency of analysis-synthesis algorithm. Constant Q filter-bank, unfortunately, can not use fast window FFT algorithms, on the other hand Q constant filter bank find their theoretical basis in the wavelet transform (time-scale transform); that can reduce

drastically the computation effort downsampling the output of each filter during the analysis, nevertheless only in dyadic case the algorithm results efficient and easy to implement.
Therefore we choose to implement a filter bank using a sampling frequency of 32 KHz, with only three constraints. First, the frequency response characteristcs, of the individual band-pass filters, are designed by frequency scaling of a prototype filter. Second, as agreed to model of the critical band, center frequencies are equally spaced across the spectrum from 0 to 500-600 Hz and with constant Q from 500 to 8000 Hz. Third, the frequency response of all the filters in parallel is maximally flat.
Using the last properties it's possible to obtain the perfect recostruction of the signal and the filter impulse response of the synthesis filter bank is then the complex coniugate and time reversed of the analysis filter impulse response.

INTRODUZIONE

Le limitazioni delle attuali rappresentazioni dei segnali in banda audio sono ben note: ci riferiamo in particolare ai suoni di alcuni strumenti musicali ed ai suoni vocalici. Queste rappresentazioni implicano in genere un modello di produzione dei suoni derivato dalla conoscenza a priori di certi meccanismi fisici degli stessi: ad esempio il modello quasi stazionario utilizzato fin dagli anni cinquanta per i suoni vocalici e' giustificato dai tempi di rilassamento relativamente lunghi nella dinamica del tratto vocale durante l'emissione dei suoni. In questo lavoro viene proposto un sistema di analisi-sintesi dei suoni musicali e della voce orientato alla rappresentazione di un modello percettivo, piuttosto che ad un modello che implichi la conoscenza del meccanismo di produzione degli stessi. Le tecniche di analisi e sintesi a tempo breve, come e' noto, forniscono un potente strumento di rappresentazione dei segnali, come la voce, permettendo di conoscere come il comportamento nel dominio della frequenza di un dato segnale evolva nel tempo. Una tale descrizione di un segnale permette di modellare, isolare i parametri di interesse ed eventualmente modificarli. L'assunzione che viene fatta per tutte queste rappresentazioni e' che il segnale da analizzare sia descritto da un modello i cui parametri si modifichino nel tempo, come si e' detto, in condizioni quasi stazionarie. In accordo con questa ipotesi le due rapprentazioni piu' popolari dei suoni vocalici e di alcuni strumenti musicali sono quelli derivati dall'analisi di Fourier a tempo breve a banda stretta e a banda larga; la prima conduce alla nota rappresentazione dei suoni mediante combinazione lineare di esponenziali complessi correlati armonicamente; la seconda, dotata di maggiore risoluzione nel tempo, porta ad una rappresentazione del segnale mediante formanti, o in generale mediante inviluppi spettrali.
Il phase vocoder e' un tipico esempio di sistemi analisi-sintesi del primo tipo; mentre la sintesi per formanti (corredato da sistemi di analisi tipo LPC o STFT) e' del secondo tipo. Un approccio alternativo nella formulazione del sistema di analisi-sintesi puo' essere affrontato,

con un modello conforme a certi comportamenti della percezione uditiva umana; in particolare un banco di filtri di analisi puo' essere progettato conformemente alla nozione di banda critica, largamente sperimentata in psicoacustica: un tale modello basato su concetti sviluppati nelle trasformazioni time-scale, puo' superare la dicotomia analisi a banda-stretta ed a banda-larga.

BASI SPERIMENTALI PER UN MODELLO PERCETTIVO

La percezione umana dei suoni musicali e della voce e' un processo alquanto sofisticato che, e' bene sottolinearlo, non e' stato compreso in tutti i dettagli. Tuttavia le esperienze consolidate relative alla percezione del pitch e la importante nozione di banda critica sono sufficienti per tentare di formulare un modello percettivo dei suoni. Infatti la teoria percettiva dei suoni e' basata su questi due concetti derivati da una serie di esperienze psicoacustiche:

a) modelli e meccanismi nella percezione del pitch.

b) il concetto di banda critica.

La teoria della percezione del pitch puo' essere di ausilio per la formulazione di un modello percettivo. Discutiamo brevemente delle tre teorie che hanno affrontato questo problema:

a) Telephone theory. (Rutherford). Questa teoria propone un modello dove lo orecchio ha una funzione passiva: e' un trasduttore che converte i suoni in segnali elettrici; il cervello rielabora questi dati grezzi ed in particolare riconosce il pitch. La teoria e' insoddisfacente perche' in contraddizione con numerosi esperimenti che dimostrano come l'informazione trasmessa al cervello sia qualitativamente differente dal segnale continuo che si presenta all'orecchio esterno.

b) Place theory. Questa teoria e' basata da una idea di Helmholtz e sviluppata da von Bekesy. Essa associa a ciascun pitch un differente punto di eccitazione della membrana basilare. Differenti frequenze stimolano preferibilmente punti diversi della membrana basilare e quindi delle terminazioni nervose. (vedi bande critiche). Questa teoria non giustifica la alta resoluzione nel percepire il pitch da parte dell'uomo.

c) Periodicity theory. (Shouten). La teoria suppone che i messaggi dalla clochea al cervello contengono piu' informazioni di quelle provenienti dalla membrana basilare. Alcune informazioni sul segnale originale sono memorizzate e mediante una tecnica di ritardi (autocorrelazione) sono percepiti i periodi (con buona risoluzione fino a 1000 hertz).

In conclusione sia la place theory che la periodicity theory hanno un loro range di validita'. Teorie piu' recenti non utilizzano quest'ultime in antagonismo ma piuttosto combinano

queste fra loro con un approccio basato sul pattern recognition. Da questo punto di vista viene enfatizzato il ruolo svolto dal cervello nell'elaborazione dei suoni, cosi' come era affermato dalla telephon theory, ma il ruolo dell'orecchio e' ben differente da quanto predetto da questa teoria; esso invia al cervello sia informazioni spettrali con risoluzione selettiva, sia informazioni temporali per la rivelazione della periodicita'. In conclusione l'orecchio fornisce al cervello due tipi di informazione: una prima informazione riguarda il tempo e la periodicita' di ripetizione, l'altra riguarda lo spettro di frequenza del suono ed e' derivata dalla intensita' di vibrazione della membrana basilare. Da queste due informazioni, probabilmente a livello del cervello, noi deriviamo le nostre senzazioni di altezza e di qualita' del suono.
Riportiamo qui una definizione, comunemente adottata come riferimento, e dovuta a Scharf: la banda critica e' quella larghezza di banda alla quale la risposta soggettiva cambia bruscamente; detta banda ha la proprieta' di variare al variare della frequenza; originariamente Fletcher aveva introdotto il termine banda critica riferendosi alla banda di rumore centrato intorno ad un tono capace di mascherare quest'ultimo. Questi autori suggeriscono che il nostro sistema uditivo utilizza un banco di filtri che sono caratterizzati dall'essere a banda stretta a bassa frequenza ed a banda larga ad alta frequenza: nella figura 1 sono forniti i valori sperimentali ottenuti per le bande critiche, utilizzati per il nostro sistema di analisi; fin da adesso possiamo notare che le bande critiche rappresentano un set di filtri in overlap piuttosto che un insieme di filtri contigui e discreti, con proprieta' che variano lungo la scala delle frequenze. Il concetto di banda critica puo' essere di grande utilita' nell'esame dello aspetto spettrale e delle sue caratteristiche in relazione al timbro. Possiamo infatti affermare che circa 50-100 componenti armoniche dovrebbero avere differenti ed importanti effetti sul timbro ma sembra essere piu' plausibile che ove piu' componenti spettrali cadano in una singola banda critica, solo la struttura combinata di essi dia un contributo alla sensazione timbrica mediante la membrana basilare. La esperienza relativa all'ascolto di due toni o di piu' toni conferma questo punto di vista.

Se due toni sono di poco differenti (battimenti fino a 6 Hz), non distinguiamo i due toni bensi' un valore medio con battimenti. Se le frequenze sono piu' distanti ma sempre nella banda critica riferita alla frequenza centrale, allora pur percependo la frequenza centrale il suono risulta modificato nell'aspetto timbrico. Analoghe esperienze condotte con modulazione di frequenza producono gli stessi risultati percettivi. Alcuni autori hanno negli ultimi anni presentato modelli psicoacustici per l'analisi e la sintesi di segnali della banda audio al fine di ottenere una efficiente compressione dei dati, senza degradazione del segnale; in particolare Johnston della Bell ha realizzato una codificazione dei suoni vocalici e musicali derivato dall'uso delle bande critiche per il mascheramento del rumore. Il nostro approccio e' piuttosto indirizzato ad una parametrizzazione e quindi una rappresentazione percettivamente

significativa tale da permettere di modificare i segnali.

MODELLO DI ANALISI SINTESI

Nella precedente sezione sono state introdotte le necessarie conoscenze di psicoacustica per realizzare un banco di filtri modellabile mediante la importante nozione, percettivamente significativa, di banda critica. Prima di procedere alla formulazione del modello, riteniamo opportuno rivedere i concetti di analisi a tempo breve e a tempo lungo, corrispondenti alle note rappresentazioni dei segnali acustici mediante somme di sinusoidi oppure mediante inviluppi spettrali, in particolare di natura formantica.
In accordo con la rappresentazione armonica, i segnali sono modellabili come una combinazione lineare di esponenziali complessi:

$$x(n) = \sum_{k=0}^{P(n)-1} c_k(n) e^{j2\pi nk/P(n)}$$

ciascuno dei quali e' prelevato da un filtro a banda stretta con ampiezza complessa e frequenza istantanea, lentamente variabili nel tempo. La rappresentazione mediante STFT di $x(n)$ puo' essere espressa mediante [Portnoff]:

$$X(n,\omega) = \sum_{k=0}^{P(n)-1} c_k(n) H(k\Omega(n) - \omega) e^{j[k\phi(n)-\omega n]}$$

nella quale $\phi(n) = \frac{2\pi n}{P(n)} = \Omega(n)n$.
Da questa si puo' notare che, se si fissa $n = n_0$, la STFT di $x(n)$ e' la somma di $P(n_0)$ immagini del filtro di analisi $H(\omega)$ ognuna traslata in frequenza di un fattore $K\Omega(n_0)$ e pesata da $c_k(n_0)e^{jk[\phi(n_0)-\omega n_0]}$.

A questo punto, nel caso dell'analisi a banda stretta, in cui si fissa la banda di $H(\omega)$ in modo che risulti minore della frequenza fondamentale di $\Omega(n)$, le immagini di $H(\omega)$ si dispongono in maniera tale da permettere la risoluzione delle singole armoniche, fornendo quindi una stima dei coefficienti impiegati nella rappresentazione armonica.
Complementarmente nel caso dell'analisi a banda larga, in cui si fissa la banda di $H(\omega)$ in modo che sia maggiore di alcune volte $\Omega(n)$, non si ottiene una risoluzione sufficiente a distinguere le singole armoniche del pitch, che invece vengono interpolate permettendo una stima dell'inviluppo spettrale del sistema.
Un sistema di analisi-sintesi che utilizzi una tale rappresentazione e', pero', pesantemente vincolato dall'ipotesi di quasi stazionarieta' dei segnali in esame: solo in tal caso esso puo' essere formulato come un sistema identita', in assenza di modificazioni dei parametri. La

Table of Critical bands, from Scharf, et. al.

Band number	Lower Edge	Center	Upper Edge
	Hz	Hz	Hz
1	0	50	100
2	100	150	200
3	200	250	300
4	300	350	400
5	400	450	510
6	510	570	630
7	630	700	770
8	770	840	920
9	920	1000	1080
10	1080	1170	1270
11	1270	1370	1480
12	1480	1600	1720
13	1720	1850	2000
14	2000	2150	2320
15	2320	2500	2700
16	2700	2900	3150
17	3150	3400	3700
18	3700	4000	4400
19	4400	4800	5300
20	5300	5800	6400
21	6400	7000	7700
22	7700	8500	9500
23	9500	10500	12000
24	12000	13500	15500
25	15500	19500	

Figure 1: Tavola delle frequence centrali e di taglio delle bande critiche

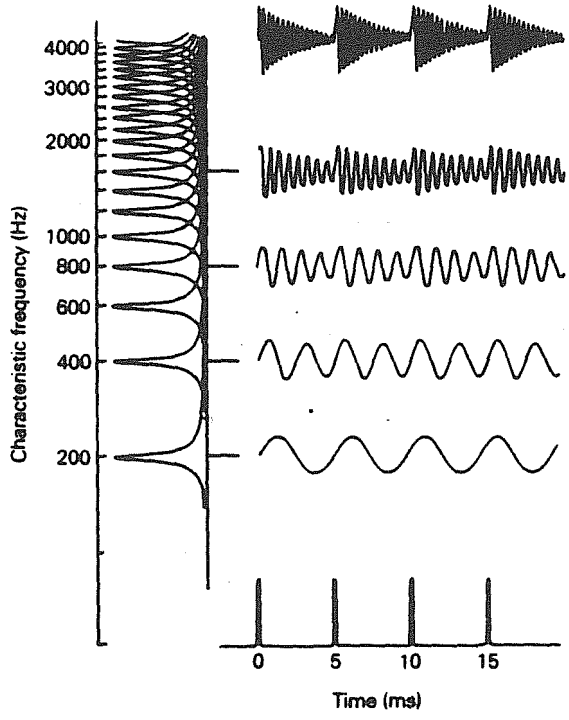


Figure 2: Modello delle bande critiche mediante banchi di oscillatori accordati

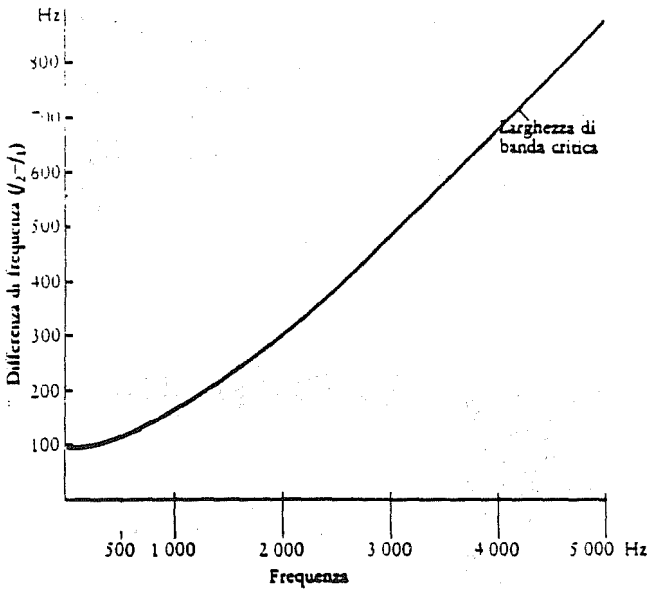


Figure 3: Larghezza di banda critica in funzione della frequenza

sintesi additiva, ben nota nell'ambito della computer music, risponde a queste caratteristiche.
Una osservazione puo' essere fatta, secondo quanto detto, circa la finestra di analisi e quindi la la banda del filtro analizzante: se la durata della finestra temporale e' molto piccola, sulla base del principio di indeterminazione, si otterra' una buona risoluzione nel tempo a discapito di quella in frequenza. Inversamente finestre di lunga durata permetteranno l'analisi dei dettagli frequenziali a spese di una scarsa localizzazione temporale. Sono del primo tipo le finestre di analisi adoperate per la stima dell'inviluppo spettrale di un segnale, specie di tipo vocalico, da lungo tempo adoperati per l'analisi della voce mediante diagrammi spettrali. La natura quasi periodica dei segnali si manifesta nella modulazione dell'ampiezza sull'asse dei tempi. Sistemi di sintesi vocale per formanti possono utilizzare questa rappresentazione per la stima dei parametri significativi.
Negli ultimi anni sono state introdotte un nuovo tipo di trasformate (time-scale e wavelet) che permettono l'uso di modelli dei suoni musicali e della voce che non siano vincolati dall'ipotesi di "quasi-stazionarieta' " che limita pesantemente la classe dei suoni modellabili. Ricordiamo brevemente alcune proprieta' delle wavelets analizzanti (che si identificano con le risposte impulsive dei filtri di analisi):

a) la wavelet ha una sua durata che dipende dalla frequenza di centro-banda del filtro passabanda considerato (la durata sara' tanto piu' piccola quanta piu' alta e' la frequenza di centro banda del canale) Una conseguenza molto importante di quanto detto il banco di filtri che costituiscono il sistema di analisi risulta non piu' a banda costante ma a Q costante cioe': df/f=const.

b) la wavelet analizzante viene traslata nel tempo con un passo che e' direttamente correlato alla frequenza del canale di analisi (passo piccolo per le alte frequenze ed inversamente per le basse).

Il comportamento dell'orecchio umano a Q quasi costante nel dominio della frequenza oltre i 500 hertz circa suggerisce l'uso delle trasformate time scale per la realizzazione del sistema di analisi-sintesi. Le bande dei filtri di analisi coincidono con le bande critiche fornite dalle tabelle, fornendo di esse una rappresentazione approssimata; come si e' detto le bande critiche sono definite nel continuo. In ogni caso, l'uso di trasformate time scale, fornisce a questa rappresentazione attributi diversi al contenuto tempo-frequenziale a secondo delle frequenze di lavoro, e in particolare la analisi a banda-stretta, e quella a banda larga forniscono una unica rappresentazione che nel caso dei suoni armonici e' di tipo sinusoidale per i canali a bassa frequenza, e mediante fasce sonore per i successivi. Piu' dettagliatamente questo punto di vista conduce ad un modello di sintesi dei segnali strumentali e vocalici che nel caso di suoni armonici, opera mediante alcune parziali (generalmente le prime tre, quattro) opportunamente inviluppate mentre per le altre si opera per fascie sonore; sono introdotti

per quest'ultime parametri derivati dall'analisi.

Il nuovo modello viene giustificato formalmente, trattando sia il caso di suoni armonici sia di suoni di natura random. Una importante caratteristica di questa rappresentazione e' la possibile modificazione dei segnali analizzati: in particolare il problema della modificazione dei parametri del modello, ad esempio la dilatazione e la compressione della scala dei tempi di segnali sonori, puo' essere riveduto sulla base di considerazioni di tipo precettivo.

IMPLEMENTAZIONE DEL SISTEMA DI ANALISI E SINTESI

Sistemi di analisi a Q costante possono essere implementati con diverse tecniche. In particolare essi possono essere realizzati come implementazione diretta della trasformata Wavelet all'interno della quale essi trovano la loro collocazione teorica oppure come banchi di filtri non uniformi realizzati con tecniche "multirate".
L'approccio mediante Wavelet integrali e' una soluzione computazionalmente pesante, che e' stata realizzata a scopo sperimentale dal gruppo ACEL a Napoli. La seconda tecnica, realizzata mediante banchi di filtri "mirror" non e' completamente versatile dal momento che tali banchi di filtri presentano un Q globalmente costante e non permettono la simulazione delle bande critiche su tutto lo spettro di frequenze. Per superare questo notevole inconveniente abbiamo sviluppato un progetto di un banco di filtri, basatosempre sulle caratteristiche "timescale" e tale da avere un comportamento molto vicino a quello delle bande critiche. La struttura globale del banco puo' essere decomposta in due sezioni: la prima, che copre l'intervallo di frequenza da 0 a circa 500 Hz, e' costituita da filtri passa-banda con $\Delta\omega$ costante, cioe' del tipo utilizzato nella STFT, mentre la seconda, che copre l'intervallo di frequenza superiore, e' costituita dal filtri passa-banda a Q costante. Descriviamo il metodo seguito per costruire la sezione a Q costante. Le risposte in frequenza dei singoli filtri del banco sono ottenute per cambi di scala, nel dominio della frequenza, secondo un parametro di scala $a \leq 1$. Fissato a la risposta in frequenza del primo filtro del banco (che sara' un passa-alto) e' costruita in modo da avere una banda passante pari a

$$\Delta\omega_1 = \pi(1 - a) \tag{1}$$

In figura 5 e' raffigurato il modulo quadro delle risposte in frequenza del primo filtro del banco e del suo complementare, nel caso $a = 0.5$.

E' intuitivo, ancorche' dimostrabile rigorosamente, che un segnale analizzato con un dato banco di filtri potra' essere ricostruito perfettamente con un opportuno algoritmo di sintesi, se e solo se dette $H_{iL}(\omega)$ e $H_{iH}(\omega)$ le risposte dell'i-mo filtro passa banda e del suo

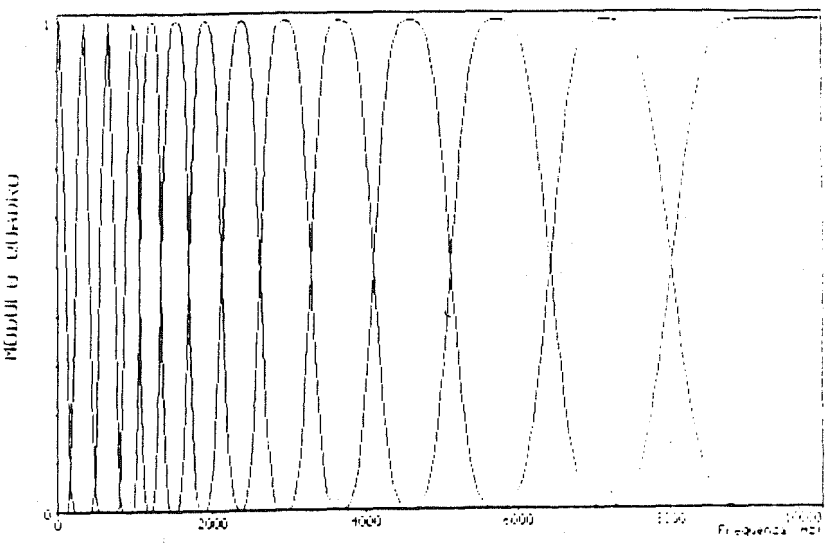


Figure 4: Un esempio di banco di filtri a Q costante

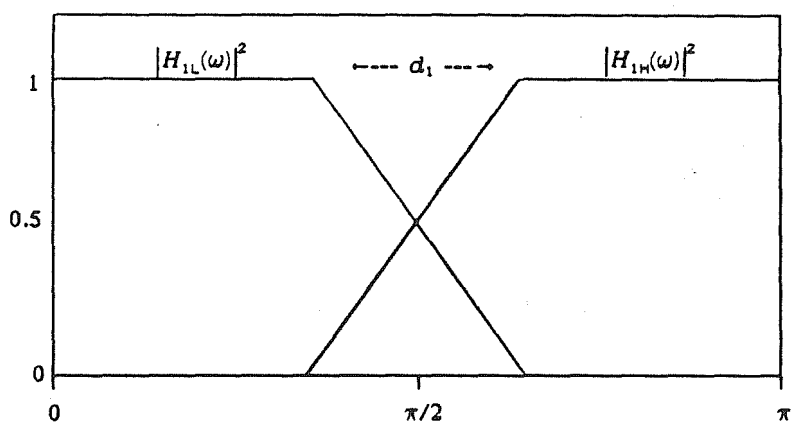


Figure 5: Modulo quadro del primo filtro del banco e del suo complementare nel caso di a=0.5

complementare, si ha per ogni i

$$|H_{iL}(\omega)|^2 \quad + \quad |H_{iH}(\omega)|^2 \quad = \quad 1 \tag{2}$$

Pertanto nel progetto del banco di filtri imponiamo il vincolo che tale relazione sia verificata per ciascun filtro passa-banda e per il relativo complementare. Cio' conduce a considerare risposte in frequenza del tipo

$$H_{iH}(\omega) = cos(\theta_i(\omega)) \tag{3}$$

$$H_{iL}(\omega) = sin(\theta_i(\omega)) \tag{4}$$

(con $i = 1, 2, ...$) che, comunque scelta la funzione $\theta_i(\omega)$ verificano la (2). Per le applicazioni che interessano, $\theta_i(\omega)$ non e' ovviamente arbitraria, ma deve essere tale che le (3) e (4) siano le risposte in frequenza desiderate. Per esempio, nel caso della figura precedente si vede che $\theta_1(\omega)$ deve verificare le seguenti condizioni:

$$\theta_1(\frac{\pi}{2}) = \frac{\pi}{4}; \theta_1(\omega) = 0 \quad per \quad 0 \leq \omega \leq \frac{\pi - d_1}{2}; \theta_1(\omega) = \frac{\pi}{2} \quad per \quad \frac{\pi + d_1}{2} \leq \omega \leq \pi;$$

in cui d_1 rappresenta la larghezza della zona di transizione del primo filtro del banco. Allora, fissati il parametro di scala a e il valore di d_1, ed essendo $\Delta\omega_1$ la larghezza di banda del primo filtro, data dalla (1) il banco di filtri e' costruito in modo che i parametri caratteristici dell'i-mo filtro passa-banda siano dati da

$$\Delta\omega_i = a^i \Delta\omega_1$$

$$d_i = a^i d_1$$

Dalla prima delle precedenti relazioni si ha, inoltre, che

$$\Delta\omega_{i+1} = a\Delta\omega_i$$

Si puo' dimostrare che il banco di filtri cosi' costruito ha un fattore Q costante che risulta fissato dal parametro di scala a. La perfetta ricostruzione del segnale e' ottenuta mediante un banco di filtri di sintesi costituito dai complessi coniugati dei filtri di analisi time-reversed.

CONCLUSIONI

Lo scopo di questo lavoro e' stato quello di formulare un modello derivato dalla percezione dei suoni e tale che fornisse una parametrizzazione significativa rispetto il contenuto di informazione dei segnali in in esame. Come viene osservato da alcuni autori con i quali siamo

d'accordo, i metodi correnti di codificazione e di parametrizzazione dei segnali digitali sono molto inefficienti dal punto di vista del contenuto di informazione. Il nostro approccio cerca di utilizzare una diversa variabile, quella psicoacustica, per comprendere e compattare l'informazione significativa; l'uso delle tecniche time-scale fornisce una rappresentazione unica che permette di rivedere alcuni concetti di codificazione e parametrizzazione. Ulteriori ricerche vanno sviluppate per verificare la validita' e per una formalizzazione piu' approfondita della nostra proposta.

BIBLIOGRAFIA

P.Basile. "Analisi tempo-frequenza dei segnali vocali", Tesi di Laurea, Univ. di Napoli, 1990.

G.De Poli, A.Piccialli and Roads. "Representation of musical signals", MIT PRESS, 1991.

G.De Poli, A.Piccialli. "Dynamic control of FIR filters for sound synthesis", Signal Processing IV, Elsevier Publ. EUSIPCO 1988.

G.Evangelista, C.W.Barnes. "Discrete-time wavelet transforms and their generalizations", Proc. Int. Symp. on Circuits and Systems, IEEE-ISCAS 90, New Orleans, 1990.

D.Gabor. "Acoustical quanta and the theory of hearing", Nature, 1947.

A.Grossman et al. "Time-scale representation obtained through continuous wavelet transform", Proc. IV European Signal Proc. Conference (EUSIPCO-88), Grenoble, 1988.

J.D.Johnston. "Transform coding of audio signals using prceptual noise criteria", IEEE Journal on Selected Areas in Communication. Vol.6 No.2, 1988.

P.Lieberman. "Speech physicology, speech perception, and acoustic phonetics" E.Blumstein Cambridge University Press, 1988.

S.Mallat. "A theory for Multiresolution signal decomposition: the wavelet representation", IEEE trans. on Pattern Analysis and Machine Intelligence, 1989.

B.C.J.Moore. "Introduction to the psycology of hearing", MacMillan Press, 1987.

M.R.Portnoff. "Time-frequency representation of digital signals and systems based on short-

time Fourier analysis", IEEE trans on Acoustics Speech and signal processing (ASSP), 1980.

M.R.Portnoff. "Short-time Fourier analysis of sampled speech". IEEE trans on ASSP, 1981.

R.Plomp. "The ear as a frequency analyzer". Journal of Acoustic Soc. of America, 1964.

L.R.Rabiner, B.Gold. Theory and application of digital signal processing. Prentice-Hall, 1975.

C.Roads, J.Strawn. "Foundation of computer music" MIT PRESS, 1985

M.J.T.Smith, T.P.Barnwell. "A new filter bank theory for time-frequency representation", IEEE Trans on ASSP, 1987.

P.P.Vaidyanathan. "Multirate digital filter banks and Polyphase Network and Applications: A Tutorial. Proc. of IEEE, Jan. 1990.

TIME-SCALE REPRESENTATIONS OF MUSICAL SOUNDS

Gianpaolo Evangelista

ACEL-Dipartimento di Scienze Fisiche,
University of Naples, Mostra d'Oltremare, pad. 20, I-80125 Napoli, Italy
E-mail : ACEL@NAPOLI.INFN.It.
Tel. +39-81-7253211

Keywords: Wavelet Transform, Filter Banks, Quadrature Mirror Filters, Subband Coding, Signal Processing, Computer Music.

ABSTRACT

The accurate synthesis of musical sounds based on the analysis of few sample tones of a given natural instrument is still an open problem. Several synthesis techniques rest either upon models of sound sources or on general signal representations which are unrelated to both the source and the listener. It may be interesting to explore algorithms which are based on models of human hearing. Time-scale representations provide a useful mathematical tool working in that direction.

1. Introduction

Combined time and frequency representations provide interesting analysis and synthesis algorithms for the processing of digital musical sounds. Probably the oldest and best known member of this class of representations is the short-time Fourier transform in which the musical sound is modeled as a superposition of sine waves with slowly varying weigths (amplitude envelopes). Additive Fourier synthesis is realized in a similar way by allowing the frequency of the sinusoidal partials to slowly change in the time. The partials of Gabor expansions are sine waves with gaussian shaped envelopes i.e. wave trains with a finite uncertainty product. Sliding window representations generalize Gabor expansions. The signal is split into several equal width frequency bands. In that case, the analysis algorithm is best implemented in a perfect reconstruction filter bank, whose outputs can be critically downsampled. The synthesis algorithm is simply given by the inverse filter bank, including interpolators. The analysis channels of the transforms so far mentioned all provide uniform time and frequency resolutions.

Recently introduced, discrete time and scale wavelet transforms are based on non-uniform band splitting. In their simplest version octave bandwidths are considered with better time resolution

at high frequencies and better frequency resolution for the lower part of the spectrum. Other versions of wacelet transforms allow better frequency resolution at the expense of time resolution. The non-uniform resolution feature of wavelet transforms has a counterpart in the characteristics of the human hearing. It may be possible to adjust the frequency channels to the critical bands of the ear. In this paper we provide an outline of discrete time and scale wavelet transforms and point out some of their applications to the analysis and synthesis of musical sounds.

2. The Wavelet Transform

The integral wavelet transform of a finite energy signal f(t), with respect to a well behaved bandpass analyzing wavelet $\Psi(t)$, is defined (Grossmann & Morlet, 1984) as the following function of two variables:

$$F(a,b) = \int f(t)\, \Psi\left[\frac{t-b}{a}\right] dt \qquad (1)$$

The variables a and b in (1) respectively represent scale and time in the transform domain. Since the analyzing wavelet is a bandpass function, to each value of the variable a it corresponds a distict frequency band. These frequency bands are generally overlapping. On the other hand, since the variable a scales the analysis time i.e. the time interval at which the wavelet is essentially different from zero, to each scale it corresponds a different time resolution. By varying the value of b one shifts the analyzing wavelet in the time by an amount which is inversely proportional to the scale. As defined in (1), the wavelet transform is both redundant and not directly suitable for digital implementations. A first discretization of (1) was obtained by exhibiting special wavelets generating complete continuous-time sets over a grid of points in the (a,b)-plane (Daubechies, Grossmann & Meyer, 1986; Mallat, 1989; Daubechies, 1988). Subsequently, the wavelet transform of sampled signals has been introduced by means of a discretization of both time and scale variables (Evangelista & Barnes, 1990). Discrete time and scale wavelet transforms can be exactly implemented in digital critically sampled filter banks without the need of analogic pre-filtering and post-filtering . In a recent paper (Evangelista & Piccialli, 1991), the definition of discrete time and scale wavelet transforms has been extended to arbitrary time and scale resolutions.

In order to define discrete wavelet transforms having roughly one octave frequency resolution one can consider the following two pairs of filters:

$$\{ G_1(z) , G_2(z) \} \quad \text{(analysis filters)}$$

and (2)

$$\{ H_1(z) , H_2(z) \} \quad \text{(synthesis filters)}$$

It is convenient to introduce the so called AC (Aliasing Cancellation) matrices:

$$\Gamma(z) = \begin{bmatrix} G_1(z) & G_2(z) \\ G_1(-z) & G_2(-z) \end{bmatrix}$$

and

$$\Lambda(z) = \begin{bmatrix} H_1(z) & H_2(z) \\ H_1(-z) & H_2(-z) \end{bmatrix}$$

The filter pairs in (2) are said to be perfect reconstruction if the following conditions are satisfied:

$$\Gamma(z)\Lambda^T(z) = \Lambda^T(z)\Gamma(z) = 2I\,, \tag{3}$$

where I is the 2x2 identity matrix.
By letting

$$\eta_{i,k}(n) = h_i(2k-n)$$

and

$$\gamma_{i,k}(n) = g_i(n-2k)\ ,\ i=1,2\ ,$$

it is easy to verify that eqs.(3) are equivalent to the following identities:

$$\sum_{j=1}^{2} \sum_{k} \gamma_{j,k}(m)\, \eta_{j,k}(n) = \delta_{m,n} \tag{4}$$

and

$$\sum_{m} \gamma_{i,k}(m)\, \eta_{j,n}(m) = \delta_{i,j}\, \delta_{n,k} \tag{5}$$

These two last conditions state the biorthogonality of the two sets of sequences:

$$\gamma_{i,k}\ ,\ i=1,2\ ,\ k=0,1,...,\ \text{the analysis set,}$$

and (6)

$$\eta_{i,k}\ ,\ i=1,2,\ k=0,1,...,\ \text{the synthesis set}\,.$$

A circuit implementing this process of biorthogonal expansion is shown in fig.1. The input signal is filtered through the analysis filter bank. The outputs of the analysis filter bank are downsampled retaining one every other sample. The sequences thus obtained represent the expansion coefficients of the input signal over the synthesis set. The input signal is then reconstructed by filtering the two sets of coefficients through the synthesis structure which consists of two interpolators. The two signals at the summing node each represent the projection of the input sequence over one of the analysis-synthesis subspaces generated by the pairs $\{ \gamma_{1,k} , \eta_{1,k} \}$ and $\{ \gamma_{2,k} , \eta_{2,k} \}$. In wavelet transforms the first pair forms a half-band low-pass channel and the second one a half-band high-pass channel. Octave band resolution discrete scale and time wavelet transforms are implemented by cascading analysis filter structures in a pruned tree diagram as depicted in fig.2. The low-pass output of each analysis section forms the input to the next analysis section. The synthesis structure is obtained in a similar fashion.

As a particular case it is possible to generate orthogonal discrete wavelet sets. This is the case when each analysis section is the transposed of the correspondig synthesis section i.e. whenever for any k we have:

$$\eta_{1,k}(n) = \gamma_{1,k}(n) , \quad i=1,2 , \tag{7}$$

or , equivalently, whenever

$$h_i(n) = g_i(-n) , \quad i=1,2 . \tag{8}$$

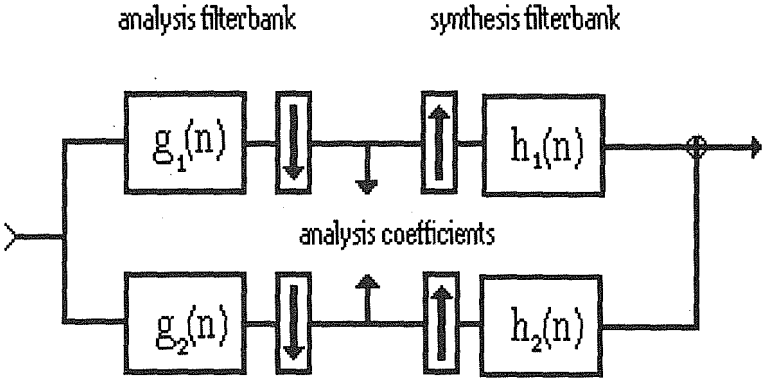


Fig.1: Analysis and synthesis structures implementing a biorthogonal expansion of the input signal.

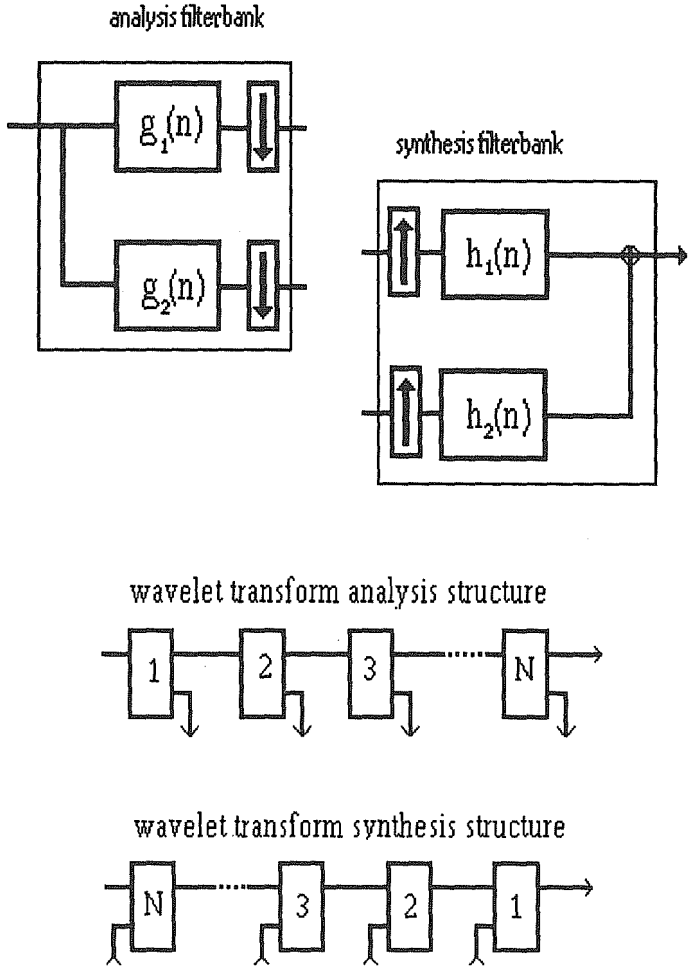


Fig.2: Discrete time and scale wavelet transform: analysis and synthesis structures.

In terms of the transfer functions identity (8) becomes:

$$H_i(z) = G\tilde{}_i(z) \, , \; i=1,2 \; , \tag{9}$$

where $G\tilde{}(z) = G^*(1/z^*)$. In terms of the AC matrix, orthonormality and completeness are equivalent to the paraunitary property:

$$\Gamma(z) \, \Gamma\tilde{}(z) = \Gamma\tilde{}(z) \, \Gamma(z) = 2 \, I \, , \tag{10}$$

where $\Gamma\tilde{}(z) = \Gamma^+(1/z^*)$, with the symbol $^+$ denoting hermitian conjugation. It is possible to show that that conditions (10) cannot be met by causal filters. However, in the case of FIR transfer functions, one can build filter banks which perfectly reproduce the input sequence within a delay. This is the case of QMFs (Quadrature Mirror Filters). An excellent review of multirate filterbanks and their design is available in the literature (Vaidyanathan, 1990). For the applications of multirate filterbanks to wavelet transforms see (Evangelista, 1989).

The wavelet analysis coefficients are obtained at the output of each leaf of the tree analysis structure. Numbering the sections in fig.2 from left to right, one obtains N scale levels. The wavelet set is defined as the set of impulse responses needed to compute the output of the tree analysis structure. From a finite depth N filter bank one obtains (Evangelista & Barnes, 1990) a set of discrete-time discrete-scale wavelets:

$$\psi_{n,m}(k) \, , \; n=1,2,...,N \, , \; m \text{ integer,}$$

the first index denoting scale and the second translation, and a set of level N scaling functions:

$$\sigma_{N,m}(k) \quad , \quad m \text{ integer} \, .$$

As already pointed out, the construction of discrete time and scale wavelet sets can be generalized to arbitrary rational time and frequency resolutions (Evangelista & Piccialli, 1991). This is particularly relevant for music analysis and sinthesis.

According to the theory of critical bands (Scharf, 1970; Johnston, 1988), the resolution of the ear averages a frequency-to-center-band-ratio of at least 0.5 . Here we shall not enter in the details of the construction of arbitrary frequency resolution wavelet sets. In the following section of this paper we shall explore some simple applications of wavelet sets to the analysis and synthesis of musical sounds and provide few examples.

3. Applications of discrete time and scale wavelet transforms

The non-uniformity of the pitch resolution of human hearing is well established. Tempered musical scales specifically reflect this behavior. Pitch detection is based on an average running over several periods and, as such, pitch is a macroscopic measure. The ability of human ear to perceive extremely short transients in great detail may appear as a conflicting characteristic. Time-scale transforms provide representations of musical sounds in which both non-uniform frequency resolution and impulsive characteristics are taken into account. In these representations the underlying assumption is that temporal resolution is sharper at higher frequencies just as frequency resolution is sharper at lower frequencies. Whether this trade-off exactly reflects the characteristics of human hearing needs to be further investigated. For the time being we shall show some examples of applications of octave-band wavelet transforms to the analysis of musical sounds. An important feature of wavelet transforms lies in their ability of separating trends from variations. Let s(n) be a sampled musical signal and consider a phase-amplitude representation of s(n):

$$s(n) = a(n) \cos \phi(n) \tag{11}$$

Amplitude and phase sequences appearing in eq. (11) can be computed by means of a Hilbert transform. The phase sequence can be unwrapped to a monotonic sequence without affecting the signal. We shall continue to denote $\phi(n)$ the unwrapped phase sequence. The amplitude envelope is a positive sequence. However, for real-world sounds, a(n) is not a smooth function and is generally rapidly varying. Let us consider a truncated orthogonal wavelet transform of the amplitude envelope:

$$a(n) = \sum_{k=1}^{N} \sum_{m} \alpha_{k,m} \psi_{k,m}(n) + r_N(n) ,$$

where

$$\alpha_{k,m} = \sum_{n} a(n) \psi_{k,m}(n) .$$

The residual $r_N(n)$ is a low-pass sequence which represents an average of a(n) over the scale indexed by N. It can be obtained by projecting the input over the scaling set $\sigma_{N,m}(k)$. If N is large enough the residual $r_N(n)$ is smooth and slowly varying and it represents the trend of a(n). The residual and some of the wavelet components of the amplitude envelope extracted from a violin tone are shown in fig.3. The components of the wavelet expansion of the envelope represent the variations from the trend, seen at different scales. The variations of the envelope become more and more refined as the scale gets smaller. This feature proves useful for changing the duration of a tone.

Similarly, one can consider the truncated wavelet expansion of the unwrapped phase:

$$\phi(n) = \sum_{k=1}^{N} \sum_{m} f_{k,m}\, \psi_{k,m}(n) + y_N(n) \quad ,$$

where

$$f_{k,m} = \sum_{n} \phi(n)\, \psi_{k,m}(n) \, .$$

In this case, the residual $y_N(n)$ represents a slowly varying phase carrier and the wavelet components represent modulants at various scales. A zoom of the phase of the same violin tone of fig.3, together with the smooth residual (in overlay) and some wavelet modulants are shown in fig.4. The last examples shows that the trend + variations feature of wavelet transforms can be exploited for FM parameters estimation.

4. Conclusions

The wavelet transform, in its digital version, provides a powerful tool for the analysis and synthesis of sampled musical sounds. It is a good candidate for the realization of listener oriented techniques. Furthermore, one can take advantage of the trend + variations decomposition provided by the transform to improve or simplify the parameter estimation procedure in several standard synthesis techniques like frequency and phase modulation.

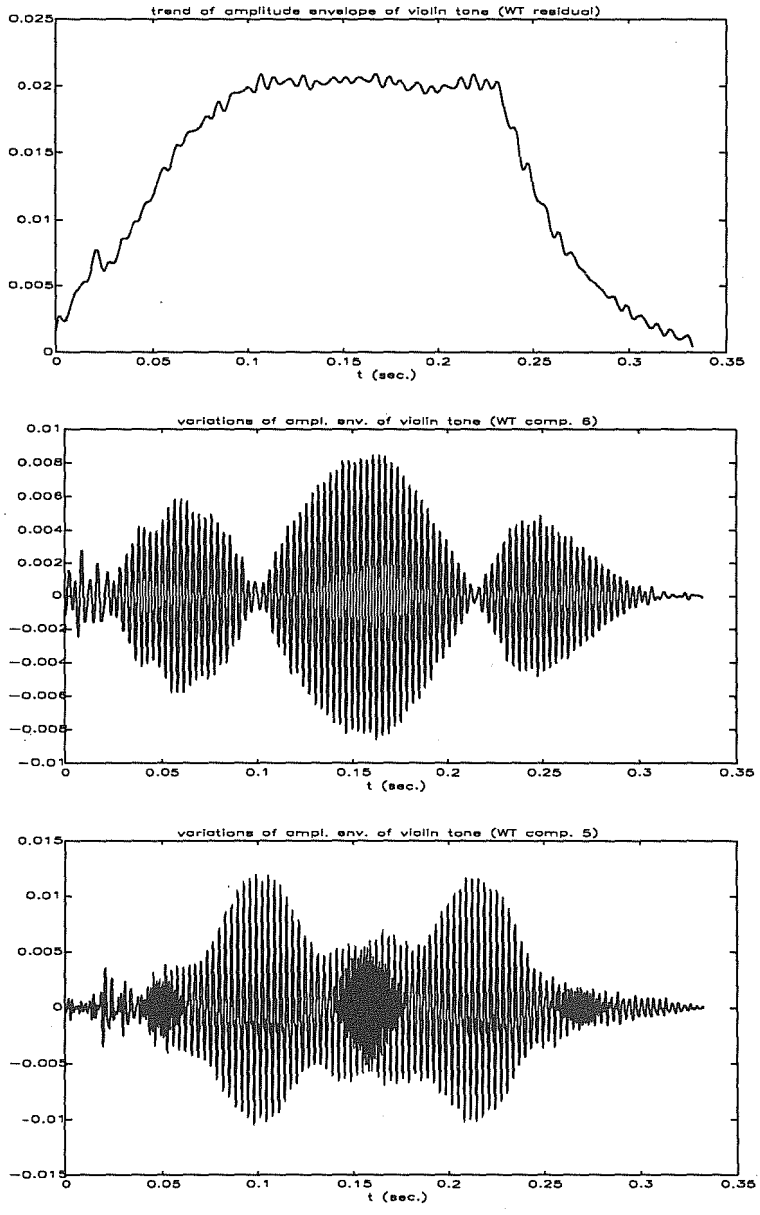


Fig.3: Analysis of amplitude envelope of violin tone via wavelet transform: trend and some of the variations.

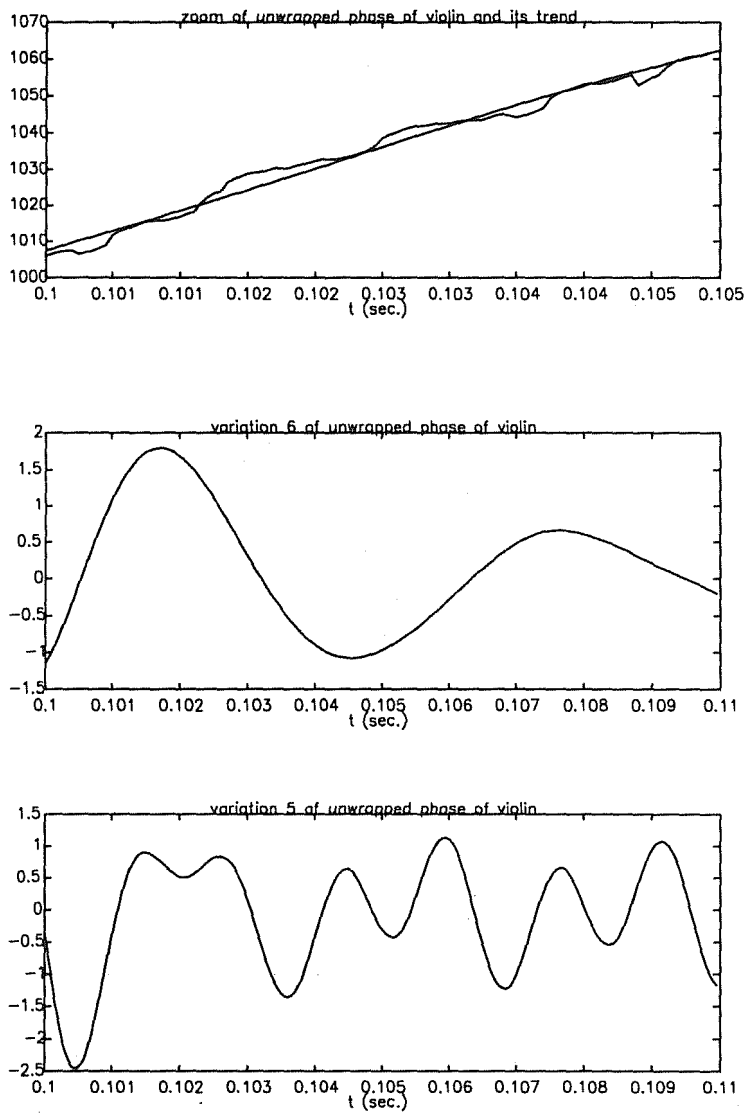


Fig.4: Analysis of unwrapped phase via wavelet transform: carrier and additive modulants.

References

A. Grossmann, J. Morlet, "Decomposition of Hardy functions into square integrable wavelets of constant shape," **SIAM J.Math.Anal.,** 15, 4, 1984.

I. Daubechies, A. Grossmann, Y. Meyer, "Painless non-orthogonal expansions," **J.Math.Phys.,** 27,(5), 1986.

S. Mallat, "A theory for multiresolution signal decomposition: the wavelet representation," **IEEE Trans. Pattern Anal. Machine Intell.,** vol. PAMI-11, no. 7, 1989.

I. Daubechies, "Orthonormal bases of compactly supported wavelets," **Comm. on Pure and Applied Math.,** vol. XLI, 1988.

G. Evangelista, C.W. Barnes, "Discrete-Time Wavelet Transforms and their Generalizations," **Proc. of the International Symposium on Circuits and Systems,** IEEE-ISCAS'90, New Orleans, La, May 1990.

P.P. Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial", **Proc. IEEE,** vol.78, pp.56-93, Jan. 1990.

G. Evangelista, "Discrete-Time Wavelet Transforms," Ph.D. dissertation, University of California, Irvine, June 1990.

G. Evangelista, "Wavelet Transforms and Wave Digital Filters," **Proc. of the 2-nd International Meeting on Wavelet Transform and Some of their Applications,** Masson-Springer, Marseilles, May 1989.

J.D. Johnston, "Transform Coding of Audio Signals Using Perceptual Noise Criteria," **IEEE Journ. on Selected Areas in Commun.,** vol. 6 no. 2, Feb. 1988.

B. Scharf, **Foundations of Modern Auditory Theory,** New York, Academic Press, 1970.

G. Evangelista, A. Piccialli, "Trasformate discrete tempo-scala," **Proc. of XIX Conv. Naz. AIA,** Naples, Italy, Apr. 1991.

CODIFICA DI EVENTI SONORI E DEI RELATIVI PARAMETRI FORMALI CON IL METODO DELL'I.F.S.

B.Fagarazzi*◆, M.Santinato*,M.Sebastiani*

*D.E.I. Dipartimento di Elettronica e Informatica.
University of Padua, via Gradenigo 6/a
35131 PADOVA ITALY
E.mail ADTPOLI@ipduniv.bitnet
tel +39(0)498287500- FAX +39(0)498287699

◆C.S.C. - Centro di Sonologia Computazionale
University of Padua

ABSTRACT

In this paper some ideas in fractal music are discussed and an IFS-based method for computer aided composition is suggested. IFS stands for "Iterated Function System", it is a powerful mathematical tool used to describe and represent any dynamical system which may exhibit any kind of attractor.
The IFS approach allows the musician to obtain good results in short time, and it makes possible also to analyze a collection of data to find a close-fitting fractal curve, for successive rielaboration. Multiple correlated parameters can be controlled too and they may be assigned to any timbral space.
A small program in C language, producing a fractal function based upon some parameters which can easily control the fractal dimension and the evolution of the curve is presented.

1.INTRODUZIONE

La nuova geometria frattale fornisce la possibilita' di descrivere fenomeni che non trovano semplice descrizione nella geometria classica (Mandelbrot, 1982) ed offre notevoli spunti alla ricerca di nuovi sistemi di composizione.
Lo scopo delle ricerche in questo campo e' di applicare questa nuova matematica per ottenere risultati esteticamente gradevoli e naturali.

Esistono molti metodi per calcolare funzioni con caratteristiche frattali.

Curve rigorosamente autosomiglianti, come quelle di Peano, di Sierpinsky, di Koch (Dodge & Bahn, 1986), possono essere ottenute partendo da un segmento, dividendo ed applicando ricorsivamnete delle trasformazioni geometriche ai sottosegmenti; il limite a cui tende la curva e', in generale, un insieme di punti che puo' avere dimensione non intera.
Esse possono essere descritte tutte da un formalismo (L-system) che si basa sull'utilizzo di grammatiche generative di Chomsky. Si tratta di un metodo talvolta utilizzato, ma scarsamente flessibile in quanto una lieve modifica dei parametri che caratterizzano la figura (le regole di trasformazione dei segmenti) porta alla generazione di un frattale con proprieta' spesso completamente diverse dalla figura di partenza.

Un altro metodo, certamente piu' noto, e' quello che fa uso di sistemi non lineari e che si realizza costruendo una mappa delle condizioni iniziali per le quali il sistema e' stabile all'infinito (detto insieme di Julia o di Mandelbrot, che ne e' una estensione). Le applicazioni al campo musicale sono di scarso interesse perche' l'algoritmo richiede notevoli risorse e i risultati sono difficilmente prevedibili, in piu' e' del tutto arbitraria la scelta del metodo che mappi la figura ottenuta in qualsivoglia spazio timbrico.

Da un punto di vista implementativo e' preferibile utilizzare un sistema non lineare (Pressing, 1988), ottenuto derivando funzioni caotiche direttamente dall'orbita descritta dalle variabili di stato, se il sistema possiede un attrattore cioe' un insieme di punti attorno ai quali l'orbita sembra convergere. I risultati dipendono strettamente dal tipo di funzione che si utilizza, e dalle condizioni iniziali scelte, per cui non puo' essere considerato un metodo universale, mentre un vantaggio notevole e' la possibilita' di implementarlo in tempo reale, per costruire ad esempio degli oscillatori caotici.

Un altro approccio e' quello di utilizzare dei frattali non deterministici, ottenuti tramite la generazione di numeri pseudocasuali, atti a generare rumore 1/f (Dodge, 1988) o algoritmi autosimili come ad esempio quello basato sul volo di Levy (Fagarazzi, 1988). Qui i problemi piu' gravosi riguardono il controllo dell'evoluzioni delle variabili, specie quando la funzione tende a fuoriuscire dall'intervallo in cui ne ha senso l'utilizzo, mentre i vantaggi principali sono la facile implementazione, anche in tempo reale.

L'Iterated Function System (IFS) rappresenta una tecnica molto potente per generare insiemi di punti o funzioni, non solo frattali ma anche descrivibili per mezzo della geometria classica, che viene normalmente impiegata nell'elaborazione di immagini.
A differenza degli approcci teste' indicati, quello che si avvale dell' IFS serve a produrre forme autosomiglianti e non, apparentemente caotiche anche se deterministiche. Un' ulteriore proprieta' significativa sta nell'elevato grado di controllo che la tecnica offre, permettendo al musicista di imporre dei vincoli compositivi anche molto stretti.
Infatti, recenti studi hanno portato alla realizzazione di algoritmi d' interpolazione e di analisi IFS che permettono di trovare i parametri ottimi per la descrizione di una qualunque forma geometrica ad una o piu' dimensioni.
In questa memoria viene presentato un metodo che differentemente da quello proposto da Gogins (Gogins, 1991) permette al musicista di stabilire a priori la dimensione frattale e

l'andamento globale della variabile da controllare. In particolare possono essere definite crescite e decrescite della funzione in un numero di intervalli scelti a piacere.
In questo modo il musicista puo' procedere, con metodo diretto e rapido alla sintesi di eventi che rispecchiano con buona approssimazione cio' che egli si prefigge anziche' dover far ricorso esclusivamente alla propria esperienza per selezionare a posteriori i prodotti piu' interessanti.
Sempre con l'IFS e' possibile ottenere famiglie di eventi con strutture simili, magari partendo da analisi di figure esistenti, semplicemente variando lievemente i coefficienti che li caratterizzano, poiche' il metodo possiede un elevato grado di stabilita'. Questa peculiarita' puo' essere sfruttata per gestire, ad esempio, le microvariazioni timbriche.

2. BASI TEORICHE DELL'IFS.

Un IFS consiste di uno spazio metrico completo (X,d) e di un insieme finito di N funzioni w_n: X -> X.

Sia B un sottoinsieme compatto di X. Si definisce spazio di Hausdorff (H(X),h) lo spazio composto da tutti i sottoinsiemi compatti di X. Si dimostra (Barnsley, 1989) che esso e' completo rispetto alla sua metrica (h), e che racchiude tutti gli elementi di H infatti, d'ora in poi col termine "frattale" si intendera' proprio un elemento di H.

Si puo estendere la funzione definita in (X,d) a tutto (H,h) in questo modo :
dato un B appartenente ad H,

$$w(B) = \{ w(x); \text{ per ogni x appertenente B} \}.$$

Si puo' ora costruire una trasformazione W:H(X) -> H(X) associata ad un IFS in questo modo:

$$W(B) = \bigcup_{i=1}^{N} w_i(B).$$

Un teorema garantisce l'esistenza e unicita' di un punto fisso per la trasformazione W, se le w_n che compongono l'IFS sono contrattive, cioe' se mappano insiemi in insiemi di area minore. In altre parole, se esiste ed e' unico un A appartenente a (H,h) tale che W(A)=A. A e' detto attrattore dell'IFS.
Si dimostra che si ottiene A=lim W(W(W.....(W(B))), qualunque sia B appartenente a (H,h).

Il teorema suggerisce quindi un metodo per calcolare l'attrattore dell'IFS: partendo da un insieme B di punti scelto a piacere si applica W calcolando l'unione delle singole trasformazioni $w_n(B)$. Iterando il processo sull'insieme ottenuto l'algoritmo si arresta quando si ottiene il punto fisso, cioe' quando l'insieme trasformato e' uguale a quello che si trasforma.
Si noti che si ottiene sempre un'approssimazione dell'attrattore come insieme di punti campionati.
Esiste un altro algoritmo per il calcolo dell'attrattore di un IFS che generalmente fornisce il risultato in minore tempo richiedendo meno risorse di calcolo. Si tratta dell'algoritmo "Random Iteration", che consiste nell'applicare ripetutamente, partendo da un punto a piacere, una delle funzioni scelta a caso al punto trasformato. Dopo un certo numero d'iterazioni si puo' considerare di aver raggiunto l'attrattore e di conseguenza, tutti i successivi punti calcolati fanno parte dell'insieme.

Normalmente vengono usate trasformazioni affini del tipo:

$$w(x) = Ax + b$$

dove x e b sono vettori e A e' una matrice,che corrisponde ad una operazione di rotazione e traslazione del vettore x di partenza.
In questo caso i parametri che descrivono la trasformazione sono 6xN ,e gia' con N=4 si ottengono degli attrattori molto complessi e interessanti.
Vediamo ora come si puo' ottenere da un IFS l'andamento che si desidera.

3. INTERPOLAZIONE FRATTALE

Si consideri il seguente problema:

Dato un insieme di N+1 punti (X0,Y0)..(XN,YN) si voglia trovare un attrattore IFS sotto forma di grafico funzione interpolatrice dei punti dati.

Si noti che l'attrattore deve essere interpretabile come funzione,e quindi non e' sufficiente che ricopra i punti assegnati.
Una soluzione (Barnsley, 1989) e' quella di usare N funzioni affini del tipo :

$$w_n\begin{pmatrix}x\\y\end{pmatrix} = \begin{pmatrix}A_n & 0\\C_n & D_n\end{pmatrix}\begin{pmatrix}x\\y\end{pmatrix}+\begin{pmatrix}E_n\\F_n\end{pmatrix}$$

e di imporre la restrizione $w_n(X_N,Y_N)=(X_n,Y_n)$ e $w_n(X_0,Y_0)=(X_{n-1},Y_{n-1})$.
Si vuole cioe' che la funzione n-esima mappi gli estremi dell'intervallo negli estremi del sottointervallo n-esimo individuato dai punti da interpolare.

In tal caso si ottengono facilmente i valori di A[n],C[n],E[n],F[n] in funzione di D[n], (che resta libero a piacere) e risultano:

A[n]=(X[n]-X[n-1])/(X[N]-X[0])
E[n]=(X[n]X[n-1]-X[0]X[n])/(XN-X0)
C[n]=(Y[n]-Y[n-1])/(X[N]-X[0])-D[n](Y[N]-Y[0])/(X[N]-X[0])
F[n]=(X[N]Y[n-1]-X[0]Y[n])/(X[N]-X[0])-D[n](X[N]Y[0]-X[0]Y[N])/(X[N]-X[0])

Il parametro D[n] influisce solo sul comportamento in "verticale" della funzione, in quanto non moltiplica la componente x. Se lo si pone uguale a zero si ottiene una semplice interpolazione mediante segmenti,(infatti le singole trasformazioni mappano tutto lo spazio in segmenti), mentre aumentandolo si ottiene sempre piu' un'amplificazione delle irregolarita' autosomiglianti tipiche dei frattali, che determinano la variazione della dimensione frattale della curva.
Da notare che vanno fissati tutti i valori D[n], per ogni trasformazione, e che quindi si puo' determinare localmente quanto deve influire sulla dimensione frattale un determinato sottointervallo. Pertanto poiche' si ottiene ogni volta una funzione rigorosamente autosomigliante, tale caratteristica si ripete ricorsivamente in ogni sottointervallo, all'infinito.
Esiste una relazione che permette di calcolare la dimensione della funzione ottenuta, se i punti di partenza non sono collineari e i D[n] non superano 1 in valore assoluto.

Infatti, se

$$\sum_{n=1}^{N} |D_n| > 1$$

allora la dimensione D dell'attrattore e' l'unica soluzione dell'equazione

$$\sum_{n=1}^{N} |D_n| a_n^{D-1} = 1$$

altrimenti la dimensione e' 1.
Nel caso di punti equispaziati la dimensione si puo' calcolare direttamente, e vale:

$$1 + \frac{\mathrm{Log}(\sum |Dn|)}{\mathrm{Log}(N)}$$

Pertanto, da quanto detto risulta chiaro che la dimensione frattale della funzione interpolante e' proporzionale alla somma dei valori assoluti dei D[n].
Una volta ottenuti i valori dei parametri delle trasformazioni affini che costituiscono l'IFS, bisogna poi procedere al calcolo dell'attrattore, che puo' essere eseguito mediante uno dei due metodi (deterministico o stocastico) sopra descritti. Per velocizzare il calcolo conviene pero' ottimizzare l'algoritmo deterministico sfruttando alcune caratteristiche della funzione interpolante.
Si puo' infatti notare che risultano noti implicitamente alcuni punti che appartengono all'attrattore. Poiche' l'interpolazione e' esatta, i punti dati da interpolare fanno parte certamente del punto fisso della trasformazione. Da essi si puo' partire per ricostruire tutto il frattale.
Un algoritmo ricorsivo, che fa uso di due array di uscita, che danno la coordinata dei punti della funzione, e' il seguente:

```
begin
   for f=0 to LEN
     X(f)=-1          ;inizializza il vettore
   for f=0 to N
     SPAWN(xx(f),yy(f))   ; trasforma gli N punti di partenza
end.
procedure SPAWN(x,y)
begin
     if( X(int(x))=-1 )  ;se il valore non esiste gia',prosegui
     begin
          X(int(x))=x
          Y(int(x))=y
          for f=1 to N
          begin
               xnew=a(f)*x + e(f)
               ynew=c(f)*x + d(f)*y + f(f)
               SPAWN(xnew,ynew)
          end
     end
end.
```

N = numero funzioni IFS
LEN = lunghezza (in punti) della funzione
xx(0..N),yy(0..N) = punti da interpolare
X(0..LEN),Y(0..LEN) = valori output della funzione,floating point.
a(1..N),c(1..N),d(1..N),e(1..N),f(1..N)=parametri di input (ottenuti tramite la formula di interpolazione)

I punti che si ottengono sono dei campionamenti non equispaziati della funzione, quindi si hanno dei valori di y per x non interi. Se si vuole una rappresentazione piu' comoda si puo' trasformare il tutto in un unico vettore di componenti Y equispaziate.
Un esempio di programma (in C) che interpola i punti dati, calcolando la funzione in modo non ricorsivo e risolve il problema sopra descritto eseguendo un'interpolazione lineare, e' il seguente:

```
#include <stdio.h>

#define MAXIFS  20
#define MAXLEN  2000
float
a[MAXIFS],c[MAXIFS],d[MAXIFS],e[MAXIFS],f[MAXIFS],yy[MAXIFS],xx[MAXIFS];
float  X[MAXLEN],Y[MAXLEN];
int    out[MAXLEN];  /* vettore funzione d'uscita */

main()
{
  float  b, x=0,y=0,xnew,ynew;
  char   s[80];
  int j;
  int sel,num;
  int  flag=-1;

  puts("numero di punti (max 20)");
  gets(s);
  num=atoi(s);
  for(j=0;j<num;j++)
  {
     printf("x[%d] ?\n",j);
     gets(s);
     xx[j]=atof(s);
     printf("y[%d] ?\n",j);
     gets(s);
     yy[j]=atof(s);
  }
  for(j=1;j<num;++j)
  {
     printf("Fattore scala vert. %d \n",j);
     gets(s);
     d[j]=atof(s);
  }
  b=xx[num-1]-xx[0];
  for(j=1;j<num;j++)
```

```
  {
    a[j]=(xx[j]-xx[j-1])/b;
    e[j]=(xx[num-1]*xx[j-1]-xx[0]*xx[j])/b;
    c[j]=(yy[j]-yy[j-1]-d[j]*(yy[num-1]-yy[0]))/b;
    f[j]=(xx[num-1]*yy[j-1]-xx[0]*yy[j]-d[j]*(xx[num-1]*yy[0]-xx[0]*yy[num-1]))/b;
    printf("%f\n0\n%f\n%f\n%f\n%f\n",a[j],c[j],d[j],e[j],f[j]);
  }
  /* stato iniziale: i punti da interpolare! */
  printf("%d\n",num);
  for(j=0;j<MAXLEN;++j)
     out[j]=X[j]=-1;
  for(j=0;j<num;++j)
  {
      printf("%d\n%d\n",(int)xx[j],(int)yy[j]);
      Y[(int)xx[j]]=yy[j];
      X[(int)xx[j]]=(int)xx[j];
      out[(int)xx[j]]=-2;
  }
  while(flag)
  {
    flag=0;
    for(j=(int)xx[0];j<=(int)xx[num-1];j++)
      if(out[j]==-2)       /* e' un punto da trasformare */
      {
        out[j]=-1;       /* non trasformarlo piu' */
        for(sel=1;sel<num;sel++)
        {
         x=X[j];y=Y[j];
         xnew=a[sel]*x+e[sel];
         ynew=c[sel]*x+d[sel]*y+f[sel];
         if(X[(int)xnew]==-1)    /* se e' gia stato raggiunto */
         {                       /* non modificarlo */
           X[(int)xnew]=xnew;
           Y[(int)xnew]=ynew;
           out[(int)xnew]=-2;  /* segnala "e' da trasformare" */
           flag=-1;
         }
        }
      }
  }
  for(j=(int)xx[0]+1;j<=(int)xx[num-1];j++)
  {
       x=(int)X[j];
       out[j]=y=(int)(Y[j-1]+(Y[j]-Y[j-1])*(x-X[j-1])/(X[j]-X[j-1]));
       plot((int)x,(int)y);
  }
}
```

Per applicazioni di tipo sonologico spetta al musicista eseguire un opportuno mappaggio del frattale ottenuto su parametri compositivi di qualsiasi tipo (associandolo per esempio ad ampiezza, frequenza, o alle varie dimensioni di qualunque spazio timbrico, oppure creando direttamente degli oscillatori con forme d'onda frattali).

E' evidente che nella maggior parte dei casi il controllo di una sola variabile non e' sufficiente a soddisfare le esigenze compositive dell'utente e si rende quindi necessario poter governare piu' parametri con la stessa metodologia. Si puo' fare cio' passando da trasformazioni 2x2 ad un caso generale nxn.

4. ESTENSIONE A PIU' DIMENSIONI

Con il passaggio ad uno spazio n-dimensionale, si ottengono contemporaneamente due vantaggi: il controllo di piu' variabili e un maggiore grado di flessibilita' su ogni parametro. Infatti, mentre nel caso bidimensionale la funzione che si ottiene e' rigorosamente autosomigliante, in piu' dimensioni puo' anche non esserlo, poiche' si utilizzano in pratica le proiezioni di una curva autosomigliante, che possiedono delle caratteristiche con piu' gradi di liberta' e quindi meno rigide e prevedibili..
Nel caso 3x3 le trasformazioni sono del tipo:

$$w_n \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} A_n & 0 & 0 \\ C_n & D_n & H_n \\ K_n & L_n & M_n \end{pmatrix} + \begin{pmatrix} E_n \\ F_n \\ G_n \end{pmatrix}$$

Per ogni variabile in piu' basta aggiungere una riga in fondo alla matrice, e una colonna alla sua destra con il primo elemento nullo, poiche' la prima riga e' legata alla variabile indipendente x della funzione interpolatrice.
Nel caso a tre dimensioni l'algoritmo opportunamente modificato fornira' in uscita tre vettori che rappresenteranno i punti della funzione interpolante nello spazio 3d. Da un IFS nxn si ottengono n-1 funzioni interpolatrici in funzione di x: si proietta cioe' (nel caso 3x3) sul piano y-x e z-x ignorando rispettivamente le componenti z e y.

I punti da interpolare sono ora del tipo $(X_0,Y_0,Z_0)...(X_N,Y_N,Z_N)$.
Imponendo le condizioni analoghe al caso bidimensionale si ottengono i valori di A[n],C[n],K[n],E[n],F[n],G[n] in funzione dei punti e di D[n]-H[n] e L[n]-M[n]. D[n] possiede lo stesso significato del caso bidimensionale ed il suo equivalente per la variabile z e' M[n].
I due coefficienti H[n] e L[n] invece forniscono un parametro per controllare rispettivamente quanto la funzione y(x) viene influenzata da z e quanto z(x) viene modificata dall'andamento di y. In questo modo si puo' ottenere una mutua dipendenza delle componenti Y-Z, permettendo di ottenere una correlazione fra i parametri.
Per il caso 3x3 i coefficienti valgono:

```
C[j]=(Y[n]-D[n]*Y[N]-H[n]*Z[N]-Y[n-1]-D[n]*Y[0]-H[n]*Z[0])/(X[N]-X[0])
K[j]=(Z[n]-L[n]Y[N]-M[n]Z[N]-Z[n-1]-L[n]Y[0]-M[n]Z[0])/(X[N]-X[0])
A[j]=(X[j]-X[j-1])/(X[N]-X[0])
E[j]=(X[N]X[j-1]-X[0]X[j])/(X[N]-X[0])
F[j]=Y[n-1]-D[n]Y[0]-H[n]Z[0]-C[j]X[0];
G[j]=Z[n-1]-L[n]Y[0]-M[n]Z[0]-K[j]X[0];
```

Non e' difficile estrarre estrapolare le precendenti espressioni al caso generale di una matrice nxn, considerando che ad ogni incremento dimensionale corrisponde l'introduzione di nuovi coefficienti dei quali 2n sono vincolati, mentre i rimanenti sono liberi.
Si ricorda che le funzioni che si ottengono sono legate alla variabile indipendente x: se si proietta la curva n dimensionale sui piani che non contengono la variabile indipendente (ad

esempio y-z per il caso a 3 dimensioni) si ottengono delle interessanti figure frattali che pero', in generale non sono funzioni, ma solamente curve nel piano.

5. CONCLUSIONI

La flessibilita' e potenza del metodo presentato lo pongono tra i piu' interessanti e meritevoli di successivi sviluppi. In particolare, sembra interessante la sua applicazione all'analisi di un evento sonoro per giungere alla sua compressione IFS.
Inoltre, data la novita' di questa teoria, unita al fascino indubbio esercitato dalle discipline frattali, ci saranno, a breve termine, altri sviluppi della base teorica che sicuramente suggeriranno nuove e piu' ingegnose applicazioni.

Biblografia

Barnsley, M.F. 1989. **Fractals Everywhere.**
New York: Springer Verlag.

Dodge,C. and C.R.Bahn.1986. "Musical fractals:
Mathematical Formulas Can Produce Musical as well as Graphic Fractals"
Byte 11(6):185-196.

Dodge,C. 1988. "Profile: A Musical Fractal."
Computer Music Journal 12(3): 10-14.

Fagarazzi, B. 1988. "Self-similar Hierarchical Processes for formal
and timbral control in composition"
Interface 17(1):45-61.

Gogins,M. 1991. "Iterated Funcions System Music"
Computer Music Journal 15(1):40-48.

Pressing,J. 1988. "Nonlinear Maps as Generators of Musical Design."
Computer Music Journal 12(2):35-46.

Mandelbrot,B. 1982. **The fractal Geometry of Nature.**
New York W.H.Freeman Co.

Capitolo 6: Musicologia

Perseo e Andromeda

opera lirica in un atto per mezzosoprano, tenore, baritono, basso e sistemi informatici di Salvatore Sciarrino

Composizione, realizzazione ed esecuzione
A. Provaglio, S. Sciarrino, A. Vidolin, P. Zavagna

CSC dell'Università di Padova
via S. Francesco, 11 - 35100 Padova - Italia

Keywords: composition, execution.

ABSTRACT

Perseo e Andromeda is a one-act opera for 4 voices and computer systems played live. It has been preceded by experiments during which the basic instruments that would produce the sound of the work were set up. The main sound synthesis algorithm is based on the subtractive principle and makes use of a state variable filter dynamically controlled in amplitude, frequency and band width. The score was written in traditional notation adding the filter opening variations and then it has been translated into MUSIC5 language to generate the samples. For live execution the sample files are played with ON STAGE! software which was specially conceived for this opera. Some parts are played in real time through 4i system. All sound are space controlled via MIDI.

L'idea compositiva

Perseo e Andromeda è un'opera lirica nel senso preciso del termine e pertanto implica innanzitutto le due proprietà caratteristiche della voce: l'intonare un testo intelligibile e l'umanità specifica del suono. Accanto ai cantanti suonano gli strumenti informatici controllati dal vivo da due esecutori.

Da un punto di vista compositivo fu necessario uno sforzo della fantasia per progettare direttamente musica di sintesi, rinunciando a mezzi belli e pronti (gli strumenti tradizionali) e formulando un mondo interamente sulle nuove macchine. Fu commissionata dallo Staatstheater di Stoccarda e lì eseguita in prima assoluta il 27 gennaio 1991.

La sperimentazione

I primi esperimenti iniziarono cercando i vari materiali sonori dell'opera partendo dai suoni del mare e del vento: l'ambiente acustico dell'isola di Andromeda. La prima preoccupazione fu quella di individuare una morfologia delle onde: il legame fra le varie grafie e i parametri di controllo del processo di sintesi.

Quest'ultimo venne naturale essere la sintesi sottrattiva e iniziammo ad esplorare le possibilità di un primo prototipo di strumento sviluppato sul Sistema in tempo reale 4i al Centro di Sonologia Computazionale dell'Università di Padova (cfr. lo schema a blocchi riportato in fig. 1). Esso consiste in un generatore di rumore bianco che viene colorato da un filtro passa basso a variabili di stato, il quale può diventare anche risonante e pertanto intonare altezze determinate sempre meno ricche di rumore fino a raggiungere un timbro quasi sinusoidale. I parametri principali di controllo sono pertanto la ampiezza globale, la frequenza di taglio e la larghezza di banda del filtro, quando diventa risonante. Le prime esperienze polarizzarono l'attenzione sulle curve di variazione di tali parametri nel tempo: curve a triangolo, sinusoidali, a segmenti di retta di varia forma. Le onde si ripetevano periodicamente e mediante potenziometri si variava il campo di escursione dei vari parametri, fissando su carta i valori delle soluzioni più interessanti e discutendo sui perché dei risultati anomali.

Durante questa fase lo strumento sintetico fu costantemente modificato e migliorato, parallelamente si imparò a 'suonarlo' scoprendone le potenzialità espressive. Molti degli esperimenti condotti furono una sorta di solfeggio finalizzato ad assimilare il mezzo e a ricreare il rapporto scrittura/suono che sta alla base della composizione musicale. Nell'arco di alcuni mesi si fu in grado di passare dall'idea sonora astratta al progetto grafico deterministico e di ritrovare all'ascolto esattamente ciò che ci si aspettava.

La partitura

La prima fase della sperimentazione terminò e si poté dunque passare alla stesura della partitura, scritta in notazione tradizionale (vedi fig. 3a) con l'aggiunta di alcuni elementi che permettessero di individuare e controllare gli strumenti già verificati e decisi in fase sperimentale.

Fino a questo momento tutto il lavoro venne realizzato con il Sistema 4i, il quale si rivelò prezioso per la possibilità di variare in tempo reale i diversi parametri dello strumento. Ma nel momento in cui si devono suonare più voci simultanee indipendenti con precise scansioni temporali e differenti andamenti delle funzioni di controllo, le azioni che un singolo esecutore può svolgere manualmente non sono sufficenti per governare la macchina. Si decise quindi di utilizzare il linguaggio Music5 per trascrivere e generare i suoni della partitura.

Sebbene la maggior parte del lavoro legato alla scrittura fosse risolto, restavano ancora numerosi problemi di carattere pratico riguardanti la trascrizione. La partitura fornisce infatti sufficienti elementi per poter procedere nella macro-orchestrazione (gli strumenti dell'orchestra sintetica da usare, le durate globali, le frequenze, le dinamiche, il valore che determina l'apertura del filtro), ma è insufficiente nella micro-orchestrazione (tempi d'attacco delle note, qualità delle dinamiche: lineari o esponenziali, distribuzione dei file nei canali di ogni singolo elaboratore ecc.). Successive fasi d'ascolto servirono a definire questi elementi. Si giunse così alla fase operativa di trascrizione che procedeva parallelamente alla stesura definitiva della partitura. In questa fase anche l'algoritmo di sintesi fu migliorato equilibrando la sua risposta dinamica in tutta la banda audio e per qualsiasi valore della risonanza.

In fig. 2 è riportato lo schema a blocchi completo dello strumento Music5 che genera la maggior parte dei suoni di sintesi. In fig. 3 è riportato un estratto della partitura scritta in linguaggio tradizionale da Sciarrino e la rispettiva traduzione per elaboratore.

L'esecuzione

Per risolvere il problema dell'esecuzione della parte informatica vennero studiate varie soluzioni; fra esse abbiamo scelto un ambiente esecutivo che nasceva dalla prassi esecutiva tradizionale. Da un punto

di vista sistemico, questa si basa sulla figura del direttore che fissa la scansione temporale e dà gli attacchi a singoli o gruppi di strumentisti i quali, una volta partiti, suonano un segmento della loro parte in maniera 'automatica' fino al prossimo punto d'incontro. Pertanto si può pensare una partitura tradizionale come un insieme di segmenti in successione che vengono attivati al momento opportuno (mediante il gesto d'attacco) e poi si sviluppano in evoluzione libera seguendo il tempo generale dell'esecuzione.

Accettando questa impostazione, l'interazione fra i mezzi informatici, i cantanti e l'azione teatrale si attua sezionando la partitura dei suoni di sintesi in una successione cronologica di file di campioni sonori che vengono fatti suonare al momento opportuno. La lunghezza temporale di ciascuno di essi dipenderà dalla musica stessa, ovvero dalla possibilità e nello stesso tempo dalla necessità di segmentazioni sempre più minute in relazione alla parte dei cantanti: se necessario si può scendere a una segmentazione a livello di singolo suono.

ON STAGE!

Per questa soluzione esecutiva venne appositamente ideato il programma ON STAGE!, una applicazione interattiva sviluppata per facilitare la riproduzione di eventi sonori in sede di esecuzione dal vivo, navigando attraverso una "partitura" fornita dall'utente, la quale descrive la sequenza di detti eventi. Allo stato attuale gli eventi possono appartenere ad una delle sole due categorie disponibili, cioè riproduzione di file audio, o sezioni di questi, e pause. Tramite un ridotto ma flessibile insieme di informazioni associate ad ogni evento è possibile specificare la durata dell'evento, l'istante iniziale nel caso dei file audio, se l'evento deve attivarsi immediatamente dopo la fine del precedente o se deve invece rimanere in attesa di conferma dall'utente o infine se l'evento in corso può essere interrotto durante l'esecuzione.

L'utilizzo tipico, in sede di concerto, è quello di avere una partitura da eseguirsi sequenzialmente, utilizzando gli eventi di tipo pausa e la possibilità di tenere l'attivazione dell'evento in sospeso per sincronizzarsi con gli altri esecutori, grazie anche ad una serie di funzionalità di cronometraggio offerta dal programma. E' comunque possibile, cosa particolarmente utile durante le prove, interrompere l'esecuzione della partitura di eventi, sospenderla, spostarsi a qualunque altro punto da cui poi riprendere, o passare ad un'altra partitura.

Allo stato attuale, ON STAGE! è più un prototipo che un prodotto finito; si è voluto infatti lasciare trascorrere un periodo di utilizzo reale, con lo scopo di verificare l'efficacia ed utilità delle funzionalità messe a disposizione, prima di passare ad una realizzazione completa. Per questa ragione, l'attenzione è stata posta più sul design legato alle funzionalità dell'applicazione piuttosto che alla interfaccia utente.

Il linguaggio adottato è il Turbo Pascal Borland 6.0; la scelta è stata dettata, in primis, dal fatto che le librerie per il pilotaggio della scheda di conversione AD/DA, messe a disposizione in via confidenziale dalla Audiologic, sono state sviluppate in tale linguaggio il quale, non producendo moduli oggetto da potersi utilizzare in un progetto a linguaggi misti, ha influito in modo determinante sulla scelta; comunque, in fase di prototipizzazione, il Turbo Pascal è stato uno strumento, per quanto limitato, rigoroso e spiccio.

Per il design e lo sviluppo è stato scelto un'approccio object oriented, il quale, pur avendo dato ottimi risultati, ha comunque sofferto della limitate caratteristiche OOP del Pascal Borland. Il nucleo della applicazione è la partitura di eventi, la quale è stata implementata come lista concatenata dinamica di oggetti polimorfi, discendenti dalla classe astratta "AudioEvent"; questa classe fornisce metodi virtuali per la attivazione, sospensione o interruzione dell'evento, nonché altri metodi informativi sullo stato dell'evento e sulle eventuali condizioni di errore. Una volta definita questa classe, la lista contenente gli oggetti di tale classe e la semplice interfaccia utente verso tali oggetti, la aggiunta delle classi denominate

"PlayEvent" e "WaitEvent", discendenti di "AudioEvent" che forniscono le funzionalità di ascolto di file audio e di pausa, non ha presentato alcun problema.

Versioni future prevedono inoltre una totale revisione dell'interfaccia utente, il passaggio ad un linguaggio ad oggetti più completo (Borland C++) e la aggiunta di funzionalità di gestione di altri eventi, principalmente legati all'interfacciamento con altri calcolatori e con apparecchiature MIDI; il design OO di ON STAGE! dovrebbe permettere l'aggiunta di questi e di altri eventi in maniera estremamente semplice ed indolore per l'architettura del sistema.

Lo spazio

Infine si progettò lo spazio sonoro, cosa alquanto complessa da farsi in astratto senza prendere come riferimento un preciso spazio reale. La maggior parte dei suoni dell'isola sono oggetti sonori che passono sopra la testa degli ascoltatori partendo da un orizzonte frontale lontano per sparire alle loro spalle. In altri casi i suoni devono avvolgere completamente l'ascoltatore dando l'impressione che arrivino da tutte le parti. Era quindi necessario pensare ad un sistema dinamico di spazializzazione che realizzasse i movimenti fissati dall'autore in partitura indicando cinque punti sorgenti del suono insieme con le traiettorie e le velocità di movimento: Horizon = orizzonte lontano; Front = boccascena; Top = sopra la testa degli ascoltatori; Rear = alle spalle del pubblico; Diffuse = avvolgente come una sfera intorno agli ascoltatori.

Anche per lo spazio si scelse una filosofia esecutiva simile a quella dei suoni di sintesi. Si stese una partitura dei movimenti che si tradusse e memorizzò su computer sezionandola in segmenti. Durante l'esecuzione vengono attivati i segmenti di spazio corrispondenti ai segmenti di suono.

Il Sistema informatico

Traducendo tutto ciò in termini di apparecchiature informatiche, per la realizzazione dei suoni di sintesi si approntarono due workstation musicali IBM compatibili basate sul processore Intel 80386/25, dotando ciascuna di esse di una memoria di massa su disco rigido SCISI da 300 MB con tempo di accesso di 16 mS e di un sistema di conversione audio digitale/analogico a 16 bit della Audiologic. Sul disco si memorizzarono i vari file di campioni sonori sintetizzati con il programma Music5. Per le parti in cui la scansione temporale degli eventi è libera si programmò opportunamente il Sistema 4i per poter variare i parametri di sintesi mediante una azione gestuale su potenziometri. La spazializzazione (movimenti e riverberi) dei suoni si affidò ad un sistema MIDI comprendente un elaboratore Atari con software Cubase, due automatic audio control della Niche e un processore di segnali SPX 1000 della Yamaha.

Ringraziamenti

Si desidera ringraziare, per i contributi durante le varie fasi del lavoro, Fabio Cappello, Sylviane Sapir, Giovanni De Poli, Graziano Tisato, Alessandro Colavizza, Roberto Cavazzana, Paolo Furlani, Audiologic s.d.f. Padova, G. Ricordi Milano.

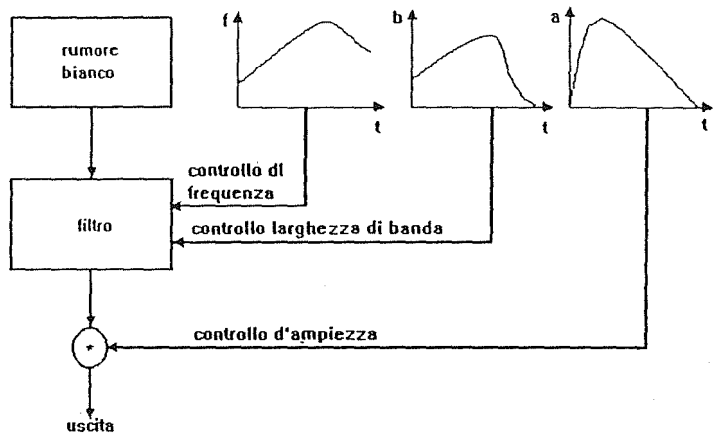


Figura 1.

OSC oscillatore
RNG forza il dato dentro un
intervallo (range)
VFM lettore di funzioni campionate
RAN generatore di rumore
STR uscita stereofonica

P5 ampiezza [dB]
P6 frequenza [Hz]
P7 incremento di frequenza [Hz]
P8 larghezza di banda del
filtro [0..8000]
P9 incremento larghezza di
banda [0..8000]
P10 funzione inviluppo d'ampiezza [F1]
P11 funzione inviluppo di frequenza [F2]
P12 funzione inviluppo di risonanza [F3]
P13 bilanciamento canale sinistro [0..1]
P14 bilanciamento canale destro [0..1]
P15 reciproco della durata dell'evento

V3 frequenza massima del rumore
V7 ampiezza massima del rumore
V9 frequenza minima
V10 frequenza massima
V11 costante di normalizzazione
larghezza di banda
V12 larghezza di banda minima
V13 larghezza di banda massima
V14 costante di normalizzazione
frequenza

F11 compensazione di ampiezza in
funzione della frequenza
F12 compensazione di ampiezza in
funzione della risonanza

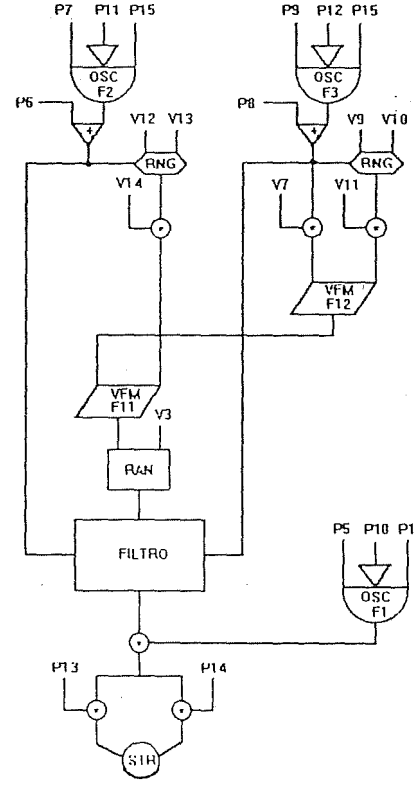


Figura 2.

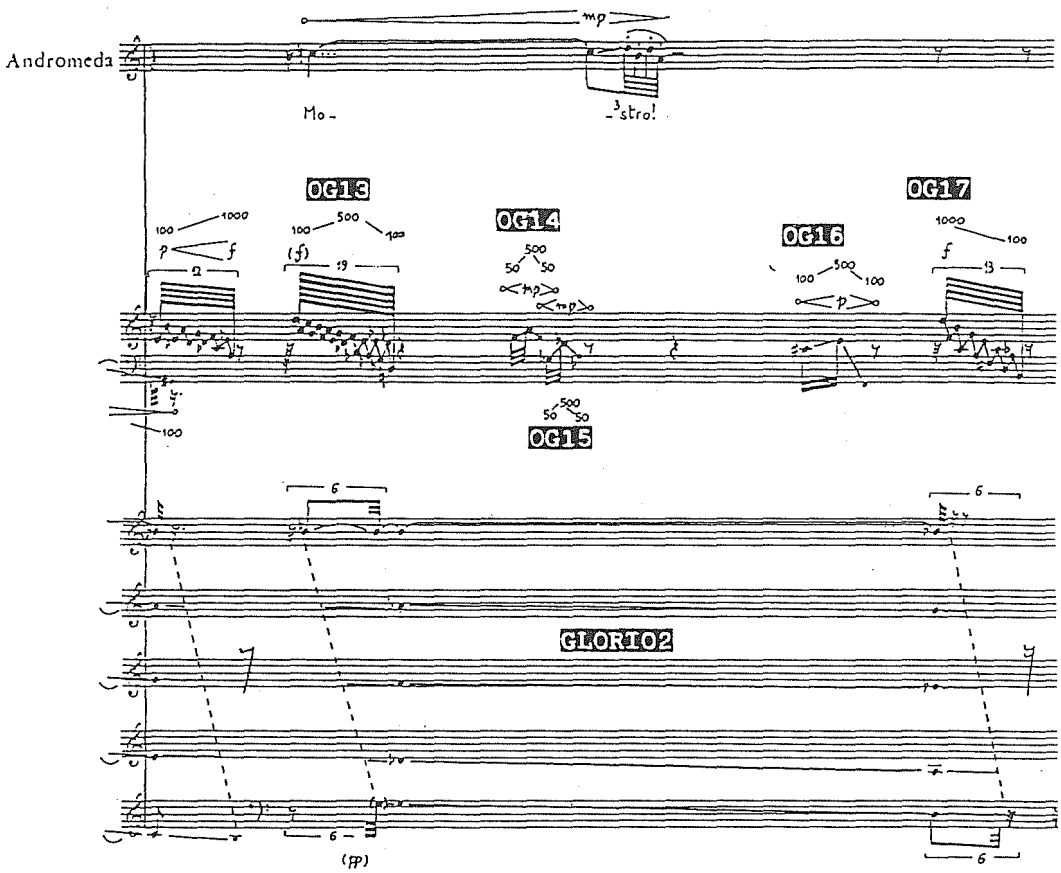


Figura 3a. Estratto dalla partitura in notazione tradizionale (pag. 10) dell'opera, per gentile concessione di G. Ricordi, Milano.

```
*************************************************************
COM PERSEO E ANDROMEDA. PAG. 10;
*************************************************************
COM A    = AMPIEZZA;
COM F    = FREQUENZA;
COM DF   = INCREMENTO DI FREQUENZA;
COM B    = LARGHEZZA DI BANDA DEL FILTRO;
COM DB   = INCREMENTO LARGHEZZA DI BANDA;
*************************************************************
COM GLORI02, 27/7/90, ORIZZONTI DA 8'18" a 8'28";
GEN 0.167 31 20 512  0,0 1,0.17 1,8.9 0,9 0,9.1;
COM ----------- A -- F --- DF -- B -- DB -- ;
NOT 0.167  1 9  55  466    0    100   0  20 17 17  1;
NOT 0.334  1 9  55  196   -49   100   0  20 17 17  1;
NOT 0.5    1 9  55  277   -57   100   0  20 17 17  1;
NOT 0.67   1 9  55  349   -38   100   0  20 17 17  1;
NOT 0.837  1 9  55  415   -13   100   0  20 17 17  1;
*************************************************************
COM OG13, 23/7/90;
GEN 0.053 31 1 512 0,0  1,0.1 1,0.847 0,0.947;
GEN 0.053 31 2 512 440,0 293,0.053 376,0.105 245,0.158
347,0.21 202,0.263 293,0.316 183,0.368 245,0.421
130,0.474 202,0.526 86,0.579 183,0.632 60,0.684 164,0.737
38,0.789 130,0.842 0,0.895 0,0.947;
COM --------------- A -- F -- DF --- B -- DB -- ;
NOT 0.053  1 0.947  75  147   440   100   400  1 2 14  0;
*************************************************************
COM OG14, 23/7/90;
COM --------- A -- F -- DF - B -- DB -- ;
NOT 3   1 0.5 65  349   91   50   450  14 14 14  0;
*************************************************************
COM OG15, 23/7/90;
GEN 3.5 31 2 512 0,0 92,0.25 23,0.5;
COM --------- A -- F -- DF - B -- DB -- ;
NOT 3.5 1 0.5 65  185   92   50   450  14 2 14  0;
*************************************************************
COM OG16, 23/7/90;
GEN 7 31 2 512 190,0 243,0.5 0,1;
COM -------- A -- F -- DF -- B -- DB -- ;
NOT 7  1  1  60   87   243  100   400  14 2 14  0;
*************************************************************
COM OG17, 23/7/90;
GEN 8.737 31 1 512   0,0 1,0.17 1,1.093 0,1.263;
GEN 8.737 31 2 512 464,0 464,0.17 226,0.247 371,0.324
139,0.401 269,0.478 110,0.554 207,0.631 52,0.708
139,0.785 24,0.862 85,0.939 0,1.016 0,1.263;
GEN 8.737 31 3 512   1,0 1,0.17 0,1.093 0,1.263;
COM -------------- A -- F -- DF --- B -- DB -- ;
NOT 8.737 1 1.263  75  123   464   100   900  1 2 3  0;
TER 10;
```

Figura 3b. Traduzione in linguaggio Music5
di un estratto della partitura (pag. 10).

STRUMENTAZIONE DI MUSICA PER STRUMENTI ACUSTICI CON SUONI SINTETICI : UN ESEMPIO APPLICATIVO, "INTERCOMUNICAZIONE" PER VIOLONCELLO E PIANOFORTE DI B.A. ZIMMERMANN.

Giovanni Cospito
Conservatorio di Musica
B. Marcello di Venezia

INTRODUZIONE : PROBLEMA, METODOLOGIA E RISULTATO.

Il seguente articolo è una introduzione teorica ad un lavoro musicale realizzato presso il Conservatorio di Musica di Venezia ed il C.S.C. di Padova dal sottoscritto e da Vincenzo Simmarano.

Il lavoro musicale si propone di utilizzare le possibilità di manipolazione dello spazio timbrico, date dai calcolatori, partendo dalle problematiche poste in ambito musicale dalla strumentazione con strumenti acustici. Viene posta anche particolare attenzione alla interazione fra i suoni sintetici ed i suoni di strumenti acustici.

La metodologia adottata parte da una simulazione convincente dei suoni degli strumenti acustici presi in considerazione, estrapola i parametri di controllo considerati efficaci ed, attraverso la manipolazione di questi, disegna spazi di variabilità timbrica.

La realizzazione musicale consiste nella strumentazione con suoni sintetici e secondo un nostro piano interpretativo, di "Intercomunicazione", un brano musicale per violoncello e pianoforte del compositore B.A. Zimmermann. Qui "strumentazione" viene intesa nel suo significato musicale tradizionale che, in questo ambito, possiamo sintetizzare con: connotare timbricamente una voce.

1. STRUMENTAZIONE: STORIA, PROBLEMATICHE, LINEE DI TENDENZA

La storia della strumentazione nella musica occidentale dal '600 ad oggi, sembra mettere in evidenza la tendenza ad assumere sempre più il timbro come essenziale elemento costruttivo della composizione musicale. Questa tendenza, parallela allo sviluppo tecnologico degli strumenti, vede inizialmente un graduale arricchimento della massa strumentale e la ricerca, almeno fino a Wagner, di un equilibrio fra la diversificazione delle caratteristiche timbriche dei singoli strumenti e la loro completa fusione nel tutti. Il timbro, comunque, viene concepito a supporto e funzione di altri parametri della strutturazione compositiva. Con Debussy e Mahler, strumentazione e timbrica assumono un valore creativo di primo piano e da allora varie sono state le intuizioni dei musicisti che hanno portato ad un interesse sempre piú accentuato per la struttura intima

del materiale sonoro.

Conviene mettere in evidenza alcune di queste intuizioni (Stroppa, 1985) perché , con il controllo assoluto e preciso del timbro tramite computer, esse possono oggi essere compiutamente realizzate e sviluppate.

Costruzione arificiale di spettri tramite la sovrapposizione di voci strumentali secondo l'ordine della disposizione spettrale, atta ad ottenere mutazioni timbriche delle voce strumentale fondamentale (Ravel).

La melodia timbrica (la klangfarbenmelodie di Schoenberg) che, ottenute nuove immagini sonore con particolari fusioni strumentali, si pone il problema di come avvicendarle con continuità e contiguità per ottenere appunto una sorta di melodia timbrica.

La tecnica delle transizioni (Webern) che tende a volersi portare con gradualità da una situazione timbrica ad un'altra, potendo percorrere traettorie diverse.

In modo simile agiscono anche le texture (Ligeti) dove il groviglio di molte note tendono a fissare una situazione timbrica globale che muta gradatamente.

Le seccessive innovazioni nella strumentazione, tentano di intevenire sulla struttura stessa del suono (Boulez): attacco, inviluppo, risonanza.

I compositori del gruppo dell'Itinèraire partono invece da una analisi del suono realizzata con mezzi scientifici avanzati per costruire le loro strumentazioni e le loro masse timbriche mutanti.

Le considerazioni conclusive sono che queste intuizioni trovano i loro limiti oggettivi negli stessi strumenti acustici che le realizzano, ma rimangono integre nella loro portata musicale soprattutto se messe in relazione alle possibilità che sono offerte oggi dalla tecnologia digitale.

2. UNA IDEA DI STRUMENTAZIONE CON SUONI SINTETICI.

Connesse alle esigenze musicali, che storicamente si sono costruite intorno al timbro, ed in relazione alle possibilità offerte oggi dal software musicale, ci preme sottolineare alcune importanti possibilità utilizzabili ai fini di una strumentazione con suoni sintetici ed, in generale, nella manipolazione compositiva del timbro:

costruzione e composizione del timbro partendo da analisi di suoni di strumenti acustici;

costruzione di timbri sconosciuti al mondo acustico usando le più comuni tecniche di sintesi;

possibilità di costruire spazi infratimbrici cosí come si parla di spazi infratonali, ed in questi spazi poter agire nella dimensione del continuo;

costruzione di sistemi di controllo parametrico ad un qualsiasi livello di astrazione e che facciano capo a qualità musicali del suono che si voglio manipolare;

micro controllo di ogni evento nella dimensione del

tempo;

acquisizione tramite campionamento di suoni concreti e loro manipolazione digitale.

La nostra strumentazione con suoni sintetici del brano di Zimmermann, parte da una iniziale simulazione integrale dei suoni degli strumenti acustici per creare innesti, con continuità, fra suoni sintetici e suoni acustici. I suoni sintetici, quindi, si muovono in uno spazio timbrico che parte dal suono acustico naturale per percorrere direzionalità timbriche differenziate. Ogni direzionalità o linea di sviluppo timbrico, viene condotta con continuità ed omogeneità all'ascolto. Si é inoltre curata una gradazione negli impasti fra le varie linee di sviluppo timbrico date al corpo del suono dei due strumenti acustici: dalla netta differenzazione e giustapposizione, alla fusione omogenea. Sono anche stati messi in macro evidenza, con varie forme di distorsione, i micro eventi interni ai suoni acustici.

3. MOTIVAZIONI DELLA SCELTA DELL'OPERA DI ZIMMERMANN E PIANO INTERPRETATIVO ADOTTATO.

L'opera di Zimmermann offriva oggettivamente la possibilità di una strumentazione con suoni sintetici basata sulla costruzione compositiva dei timbri, perchè essa stessa è costruita sulle variazioni interne alla struttura dei suoni dei due strumenti acustici e sulla interazione delle loro rispettive risonanze. Valga per esempio l'uso del pedale di risonanza del pianoforte che azionando brevi tocchi degli smorzatori sulle corde vibranti di un accordo causa variazioni del contenuto spettrale dei suoni.

Le parti dei solisti sono state lasciate generalmente inalterate. L'intervento dei suoni sintetici cresce gradatamente attraverso tutta l'opera: dai pochi interventi nella prima parte (batt. 1-348) che danno apertura spaziale alle risonanze del pianoforte, agli interventi a tre livelli (risonanze del pianoforte, spazio infratonale del violoncello, suoni armonici dello stesso violonvello) nella seconda parte (batt.349-504), fino alla orchestrazione ampia della parte pianistica nella sezione finale (batt. 715-777), dove le risonanze del pianoforte confluiscono in una coralità pseudo vocoidale conclusiva dell'intervento sintetico.

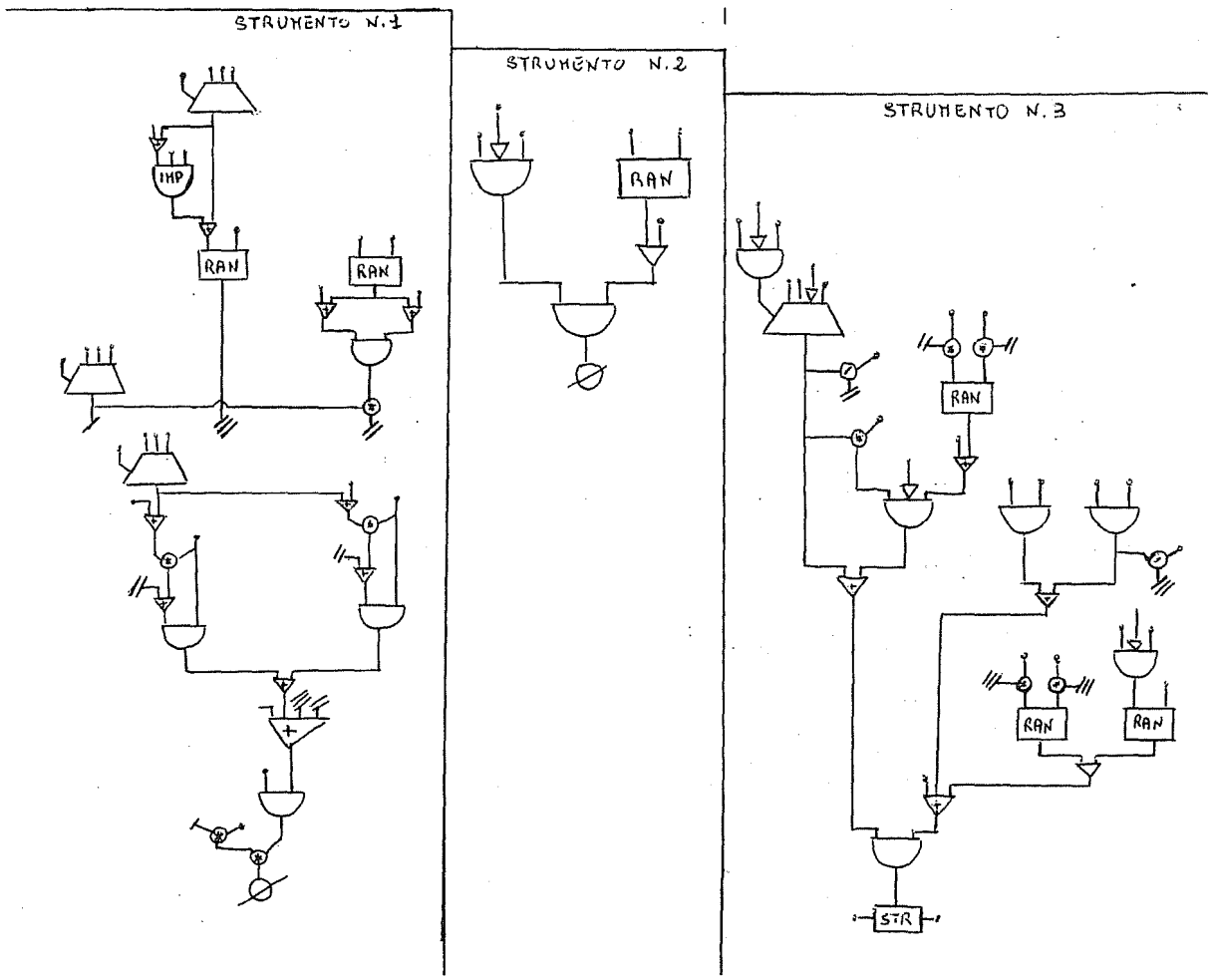
4. TECNICHE SOFTWARE.

Per lo sviluppo degli algoritmi di di sintesi è stato usato il Music 5 di Max V. Mathews (Mathews,1969). Sono stati messi a punto vari strumenti Music 5 e le tecniche di sintesi usate sono state: l'additiva; la modulazione di frequenza semplice, aleatoria e con più portanti; l'additiva di modulazione di frequenza semplice.

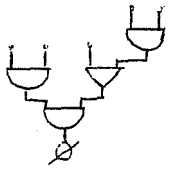
Il modello fondamentale seguito nella costruzione degli

algoritmi di sintesi è quello proposto da Morrill (Morrill,1977) dove oltre alla modulazione di frequenza a più portanti con opportuni controlli sugli indici e sui picchi d'ampiezza delle singole portanti, vi è una intera ed articolata parte per il controllo del vibrato, del portamento e del rumore che va ad agire sulla variazione nel tempo dei parametri che producono lo spettro nella modulazione di frequenza.

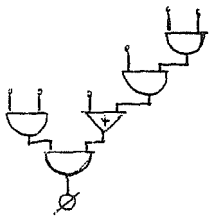
Lo strumento Music 5 n.1 simula/varia il violoncello; lo strumento n.2 è usato in risonanza agli armonici del violoncello e per distorcere opportunamente l'attacco percussivo dei suoni del pianoforte; lo strumento n.3 è usato per la simulazione/variazione del suono del pianoforte in sintesi additiva; gli strumenti dal n.4 al n.8 si innestano sulle risonanze del pianoforte e seguono una legge di transizione graduale aspro/dolce; lo strumento n.9 esegue la parte vocoidale finale.



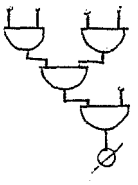
STRUMENTO N.4



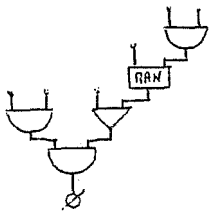
STRUMENTO N.5



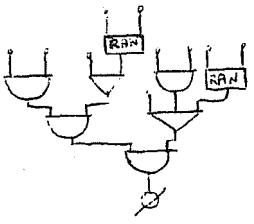
STRUMENTO N.6



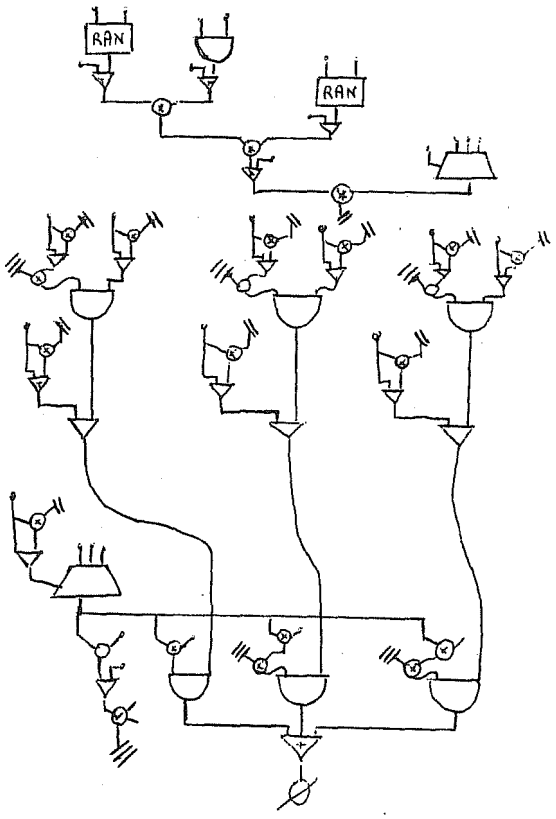
STRUMENTO N.7



STRUMENTO N.8



STRUMENTO N.9



5. SOLUZIONE ESECUTIVA.

La soluzione adottata è stata l'uso di due PC in parallelo, con due convertitori DA stereo, che inviano in esecuzione all'enter, in forma incrociata, i files audio, permettendo oltre alla perfetta sincronizzazione con i solisti, la possibilità di rispettare le agogiche tipiche delle esecuzioni dal vivo. Viene posta cura nella disposizione spaziale degli altoparlanti alcuni dei quali sono posti sotto la cassa armonica del pianoforte ed in vicinanza del violoncello per dare una giusta messa a fuoco anche spaziale della continuità fra suoni acustici e suoni sintetici.

BIBLIOGRAFIA

(I) Schoenberg, Arnold, Manuale di Armonia, Milano, Il Saggiatore,1963 (edizione originale: Schoenberg,A., Harmonieliehre, Universal Edition, Vienna, 1922);

(II) Mathews, M. V., The Technology of Computer Music, Cambridge, Massachusetts: MIT Press, 1969;

(III) Risset,J.C., An Introductory Catalog of Computer synthetized Sounds, Bell Laoratories, Murray Hill, New Jersey, 1969;

(IV) Casella A., Mortari V., La tecnica dell'orchestra contemporanea, Milano, 1973;

(V) Chowning, John M., The Synthesis of Complex Audio Spectra by Means of Frequency Modulation, Journal of the Audio Engineering Society 21(7), 1973;

(VI) Rognoni, Luigi, Fenomenologia della musica radicale, Milano, Garzanti, 1974;

(VII) Erickson, Robert, Sound Structure in Music, University of California Press, 1975;

(VIII) Morrill, Dexter, Trumpet Algorithms for Computer Composition, Computer Music Journal 1(1): 46-52, 1977;

(IX) De Poli, Giovanni, Tecniche numeriche di sintesi della musica, Bollettino LIMB n. 1 , la Biennale di Venezia, 1981;

(X) Risset J., Wessel D., Indagine sul timbro mediante analisi e sintesi, Bollettino LIMB n. 2, la Biennale di Venezia, 1982;

(XI) McAdams, Stephen, Spectral Fusion and the Creation of Auditory Images, in Music and Brain: The Neuropsychology of Music, Manfred Clynes, Plenume Press, New York, 1982 (traduzione italiana in Bollettino LIMB n. 2, La Biennale di Venezia, 1982);

(XII) Stroppa, Marco, L'esplorazione e la manipolazione del timbro, Bollettino LIMB n. 5, la Biennale di Venezia, 1985.

CAOS DETERMINISTICO, COMPOSIZIONE E SINTESI DEL SUONO

Agostino Di Scipio

CSC - Centro di Sonologia Computazionale
Università di Padova, Via S. Francesco 11, I-35121, Padova, Italy
tel. (39) 49 8283755 - fax (39) 49 8283733

Via Salaria Antica Est 33a, I-67100, L'Aquila, Italy
tel. (39) 862 313847

Keywords: composition, granular synthesis, timbre, deterministic chaos, non-linear dynamics, symbolic vs. subsymbolic treatment.

Abstract

Mathematical models of non-linear (n.l.) dynamic systems are employed in many fields of science and research, and recently came of interest for various composers using computer technology. Since effective controls over the continuum from order to disorder suggest a variety of musically interesting results, such models seem to be appropriate practical tools for compositional purposes. They show alternatively ordered and unpredictable behaviours, and are analytically non-integrable even if generated by strictly deterministic methods. What's most important from an operating point of view, all information related to a specific n.l. system is caught by the values of its starting conditions, no matter how complex the way the system behaves.

The typical features of some models, called *one-parameter families of iterated maps in a given interval*, involve multiple levels of organization, parameter-dependence (*sensitivity*) and other interesting properties. In the compositional applications presented, the basic principle is mapping the interval of a n.l. system onto an array of given elements. In the case of macroscopic organization of conventional musical parameters, such principle can often result in arbitrary mappings, not really different from other kinds of stochastic distributions; but it seems inherent and effective in the case of microscopic (phase level) controls of complex sonic event, making available particular audio domain behaviours as related to non-linearity and self-organization of natural systems. Various musical procedures are discussed, as used in the author's compositional experiences. Microscopic applications involve granular synthesis of sound and other signal processing algorithms.

Some final considerations indicate these methods as a means of *subsymbolic* approach to music composition, the strategies being concerned with the structure and the dynamic morphology of sound material (growing up to the state of musical symbols) rather then with the formal symbolic organization in pitch and duration and other high-level musical parameters.

1. Introduzione

"Music, in its final appearance [...], preserves at least traces of the processes by which it emerged from chaos"
(H. Brün, 1966)

Negli ultimi decenni molte categorie della dinamica classica hanno subito profonde riformulazioni, alcune delle quali dovute all'osservazione che sistemi dinamici perfettamente deterministici ed apparentemente piuttosto semplici manifestano comportamenti di enorme complessità, spesso irriducibili e caotici, ma che esibiscono nondimeno forme di organizzazione molto particolari. Recenti contributi ne hanno studiato i relativi modelli matematici (Collet & Eckmann, 1980; Carmona & Nualart, 1989), nella cui forma i sistemi dinamici non-lineari sono divenuti di interesse concreto in molteplici discipline scientifiche, ma anche nelle arti visive - cinematografia, *computer graphics* (Dewdney, 1991) ed in esperienze di *computer music*.

Nelle soluzioni presentate in questo articolo, modelli simili sono adoperati per specifici intenti compositivi, e nelle forme acquisite in concrete realizzazioni musicali dell'autore. In particolare è interessante poter collegare le proprietà di auto-organizzazione di tali sistemi alla progettazione della superficie acustica e, in definitiva, del timbro (Di Scipio, 1990); altri studi appaiono dedicati soprattutto ad organizzazioni formali legate all'altezza ed alla durata del suono (Pressing, 1988; Degazio, 1986).

Si osserveranno alcune proprietà notevoli dei sistemi dinamici non-lineari, fornendo un esempio di composizione algoritmica fondata su simili procedure. Successivamente si analizzeranno applicazioni relative a procedure di sintesi digitale del suono, in cui le proprietà dei sistemi non-lineari sono condotte a livello della fase del segnale acustico. Infine saranno formulate alcune considerazioni sul senso e sulle implicazioni musicali inerenti alle procedure analizzate.

2. Proprietà dei sistemi dinamici non-lineari

Nella forma più semplice, il modello di un sistema dinamico si presenta come l'iterazione

$$x_{n+1} = f(x_n) \qquad \text{[eq.1]}$$

dove il valore x_n calcolato alla n-esima iterazione della funzione f, viene considerato a sua volta argomento della stessa funzione. La eq.1 rappresenta anche un generico modello di sistema non-lineare la cui evoluzione (che evidentemente dipende da f), varia in modo sostanziale al variare del valore iniziale, cioè al variare di x_0. Esiste uno specifico *orizzonte temporale* (Prigogine & Stengers, 1988) oltre il quale la evoluzione di un sistema non-lineare è imprevedibile, pur essendo noti sia lo stato attuale x_n che f. In cibernetica questo comportamento denota una "macchina non banale": dal risultato conseguito, e conoscendo il dato in ingresso, non si può inferire la funzione di trasferimento (von Foerster, 1985). Forme di non-linearità e di auto-organizzazione come quelle che caratterizzano i sistemi dinamici caotici sono tipiche dei sistemi viventi (Maturana & Varela, 1980).

Iterazioni monoparametriche

Saranno presi in considerazione sistemi definiti da iterazioni del tipo

$$x_{n+1} = 1 - r\,x^2_n \qquad \text{[eq. 2]}$$

con *r* unico parametro del sistema; i modelli considerati, pertanto, sono *iterazioni monoparametriche*. Per *r* costante, differenti valori in x_0 causano evoluzioni differenti; la eq.2 va intesa come la forma generale di innumerevoli casi specifici, poiché x_0 può essere un numero reale qualsiasi nell'intervallo [0,1]. L'intero sistema è invece caratterizzato dal diverso comportamento dovuto al variare di *r*. In altre parole, *la successione* dei valori x_n dipende da x_0 ma il *tipo di evoluzione* è funzione di *r*.

L'evoluzione di un sistema dinamico non-lineare può:
- assestarsi su un valore di attrazione stabile (*punto fisso)*;
- assestarsi su una serie di valori, o ciclo di attrazione stabile *(ciclo limite)*;
- alternare fasi di evoluzione caotica e fasi di attrazione instabile;
- avere un comportamento apparentemente del tutto arbitrario, pur entro i limiti del suo *spazio delle fasi* (l'intervallo entro il quale avviene l'iterazione).

Nei primi due casi il valore di x_0 è irrilevante in rapporto al destino dell'evoluzione che ne deriva, tendente a stabilizzarsi su un punto fisso o su un ciclo limite. Se invece punti fissi e cicli limite sono instabili, si manifesta una profonda irregolarità, pur sussistendo saltuariamente l'azione di eventuali attrattori. In questi casi a variazioni infinitamente piccole di x_0 corrisponderanno sostanziali cambiamenti nella successione x_n. Questa proprietà è detta *dipendenza sensibile dalle condizioni iniziali* (Collet & Eckmann, 1980:15 e sgg). Per esempio, l'iterazione

$$x_n = (1 + r)\,x_{n-1} - r\,x^2_{n-1} \qquad \text{[eq. 3]}$$

si comporta diversamente per valori diversi di *r* (fig. 1). Nelle zone più caotiche (*r*=3) gli estremi dello spazio delle fasi del sistema costituiscono due punti fissi instabili, ed il sistema mostra un comportamento *ergodico*: dato un numero infinito di iterazioni, tutti gli infiniti punti dello spazio delle fasi verranno visitati. Gli istogrammi in fig. 1 (accanto al grafico di ogni evoluzione) forniscono una informazione immediata su queste proprietà, e rendono osservabile il tipo di distribuzione delle probabilità in specifiche condizioni evolutive. (La eq. 3 venne introdotta durante lo scorso secolo dal sociologo tedesco P.F. Verhulst come modello della crescita di intere popolazioni in un data nicchia ambientale; cfr. Gleick, 1987: cap. III).

Per osservare il comportamento d'insieme di un sistema non-lineare, si fa riferimento alla cosiddetta *mappa di biforcazioni*, calcolata mediante successive inizializzazioni di un medesimo modello iterativo, con *r* crescente ed x_0 fisso. In fig. 2 si osserva come nel sistema definito da

$$x_n = r\,x_{n-1} - (1-x_{n-1}) \qquad \text{[eq. 4]}$$

a differenti valori di *r*.(sull'asse verticale) corrispondano differenti evoluzioni nello spazio delle fasi del sistema. Il grafico presentato è relativo a $[r', r''] = [2.5,4]$ con $x_0 = .5$; sono evidenti i successivi *punti di*

fig. 1 - Diversi comportamenti di un sistema dinamico non-lineare in funzione del diverso valore del parametro *r* (fattore di crescita)

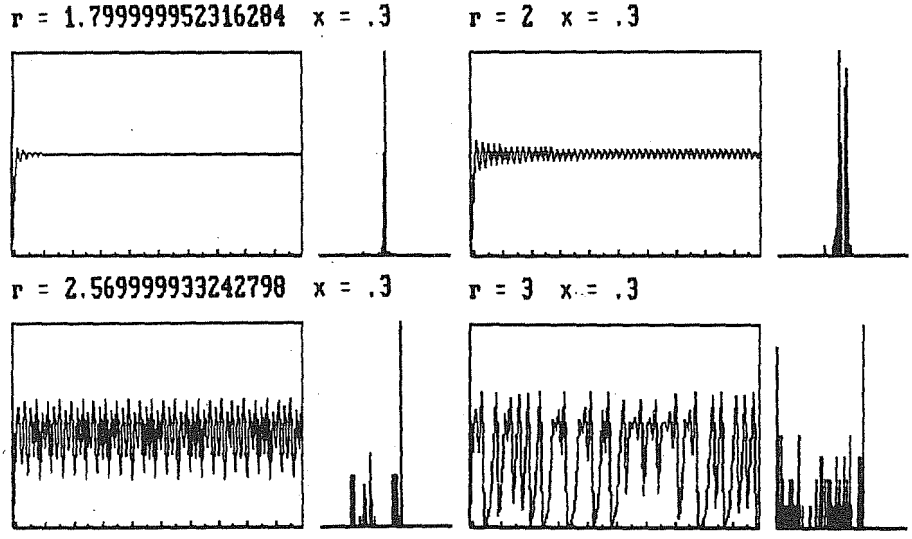
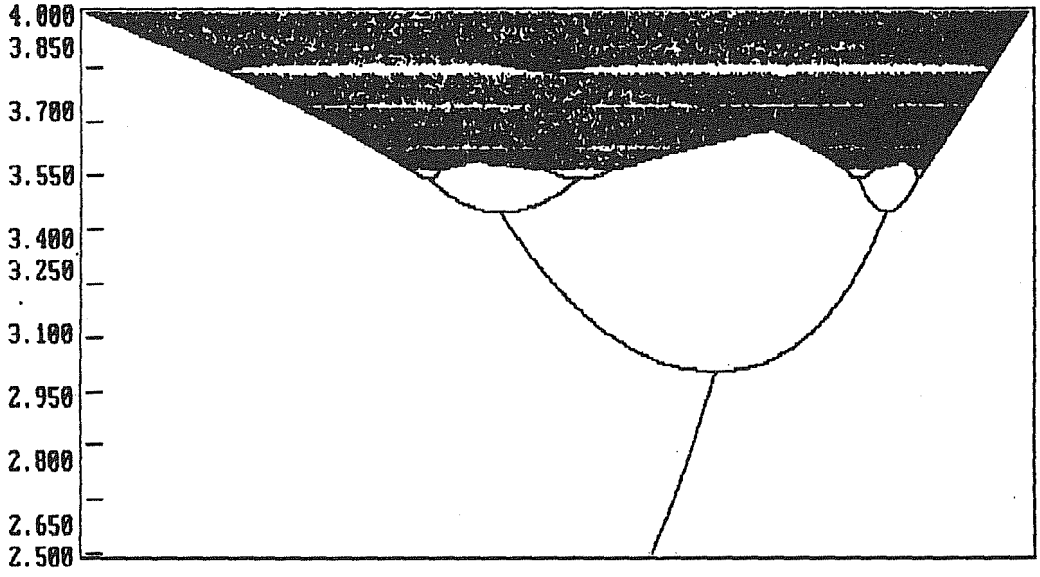


fig. 2 - La mappa di biforcazioni illustra il comportamento complesso di un sistema dinamico non-lineare il cui modello si riduce ad una iterazione monoparametrica; al variare di *r* (asse verticale) muta l'evoluzione del sistema, tendente a 0 nella regione definita da [*r'* , *r''*] = [0 , 1] ad un punto fisso [1 , 3], ad un ciclo limite-2 [3 , 3.449], ad un ciclo limite-4 [3.449 , 3.544] o a condizioni imprevedibili e confuse; anche nelle zone caotiche [3.556 , 4] compaiono saltuariamente degli attrattori ciclici (anche dispari).



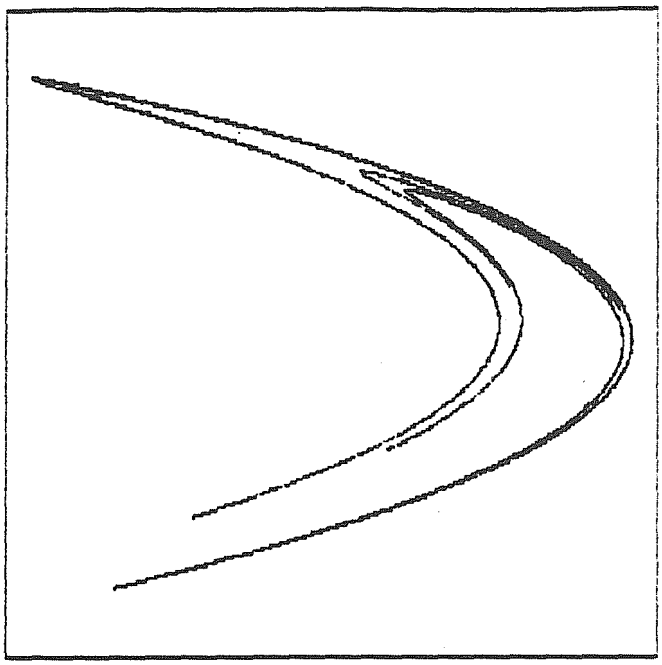


fig. 3 - L'attrattore *strano* di Hènon, una struttura frattale generata da una iterazione biparametrica

biforcazione (o di *transizione di fase*), cioè i valori di *r* per i quali muta l'evoluzione del sistema (raddoppiamenti del periodo di cicli limite e regioni caotiche). La mappa di biforcazioni è un oggetto di dimensione frazionaria, ed inoltre è *autosimile*: ciò che nell'intera mappa appare come una singola linea di rivela, in successive riduzioni di scala, una minuscola e completa replica della struttura generale in ogni minimo dettaglio.

Un esempio di iterazione biparametrica

Proprietà analoghe a quelle dei sistemi definiti da iterazioni monoparametriche possono essere riscontrate anche in iterazioni bi- e tri-parametriche. Per brevità citeremo solo il sistema a due parametri da cui emerge il cosiddetto *attrattore di Hénon* (Gleick, 1987:cap. V), di forma assai particolare (fig. 3); il modello è definito in modo completo dalla iterazione

$$x_n = y_{n-1} + 1 - r\,x^2_{n-1} \;;\; y_n = .3\,x_n \qquad \text{[eq. 5]}$$

con $r = 1.4$. Nonostante l'attrattore, il sistema è del tutto caotico, nel senso che i valori x_n ed y_n compaiono in successioni sempre diverse per ogni minima variazione di x_0 e di y_0. L'attrattore di Hénon è infatti un ciclo limite instabile ed estremamente complesso; esso è di dimensione frazionaria (ma non è strettamente autosimile), ciò che lo connota come attrattore *strano*.

Un principio applicativo generale

La dialettica che oppone ordine e disordine, determinismo ed alea nei sistemi dinamici non-lineari, è di grande suggestione. Da un punto di vista pratico, rilevante per le soluzioni musicali che presenteremo, l'interesse primario è dato dalla quantità di possibilità differenti racchiuse in modelli iterativi piuttosto

semplici, e dalle forme di organizzazione che ne emergono, che alternano strutture ridondanti (attrattori, regolarità, ripetizioni) ed imprevedibili (variazioni incessanti, irregolarità, rumore). E' anche molto importante che l'informazione relativa ad un dato sistema sia cristallizzata in soli due valori (per le iterazioni monoparametriche); ciò permette di replicare strutture molto complesse (non assimilabili ad alcuna funzione statistica nota) nei più minimi dettagli, per quanto imprevedibile ne sia l'evoluzione.

Il principio generale adottato è quello di riscalare l'intervallo di iterazione di un dato sistema dinamico non-lineare in modo che coincida con lo spazio discreto di un vettore *V* di elementi numerabili. Il valore x_n, che descrive lo stato istantaneo nell'evoluzione del sistema, coincide allora con uno degli elementi *V(i)*, cioè *lo seleziona*. Poiché x_n è in teoria un numero reale (e praticamente una sua rappresentazione ad 8 bytes, nella implementazione su computer), il riscalamento di x_n implica un arrotondamento all'intero corrispondente all'indice di *V(i)*, e quindi neutralizza di un certo numero di cifre decimali (queste però non possono essere trascurate nel calcolo della iterazione). Per minimizzare questa inevitabile distorsione, e per tracciare con buona approssimazione l'evoluzione del sistema adoperato, è perciò preferibile gestire un numero di elementi quanto più grande possibile.

Sulla composizione di "fractus"

E' possibile adoperare i modelli dei sistemi dinamici non-lineari come strumenti di composizione assistita da elaboratore. Dato un modello ed un insieme di elementi musicali *V*, l'organizzazione formale del lavoro si determina in funzione della selezione attuata in *V*. Nella realizzazione di *"fractus"* (per viola e suoni generati mediante elaboratore - 1989/90), x_n è stato riscalato su un insieme numerato di elementi musicali caratterizzati come comportamenti timbrici particolari, cioè già considerabili come *simboli* nella articolazione dell'intera costruzione. Questo tipo di applicazione macrostrutturale sarà denotata, più avanti, come procedura di *trattamento simbolico*. La partitura è generata da successive esplorazioni nella regione della mappa di biforcazioni $[r',r''] = [3.862 , 3.9996]$ dell'iterazione

$$x_n = r\, x_{n-1} - (r\, x^2_{n-1}) \qquad \text{[eq. 6]}$$

Ricorsivamente ogni sequenza di simboli viene considerata come un nuovo vettore di riferimento, nel quale la eq. 6, una volta aggiornati i parametri, realizza la successiva selezione. Ogni selezione genera i simboli che realizzano gli eventi sonori di una singola sezione del brano. Da questa procedura ricorsiva, che infine si focalizza su un ristretto numero di simboli, dipende anche la parte dei suoni realizzati mediante computer e registrati su nastro, il cui intervento è dato dall'occorrenza dei due punti di attrazione instabili corrispondenti agli estremi dello spazio delle fasi. (Per la sintesi del suono si è usato il linguaggio MUSIC360, con un algoritmo di FM a tripla portante, dotato di vari controlli. Di buona efficacia percettiva è stata la realizzazione di glissandi e di leggeri spostamenti in frequenza prodotta da un *movimento browniano* - rumore $1/f^2$ - per passi unitari di ampiezza proporzionale ad 1/10000 delle frequenze portanti).

La struttura finale di *"fractus"* è assimilabile ad un *processo direzionato*, ma è stato praticamente impossibile prevederne (e tanto più determinarne) in sede compositiva la reale direzione ed i tempi di evoluzione. Ciò si traduce in una condizione percettiva importante nell'ascolto del brano: non vi è *sviluppo* in senso convenzionale - la procedura produce una certa naturale progressività che con metodi perfettamente deterministici (cioè esplicitamente finalizzati e guidati) sarebbe stato difficile ottenere. D'altra parte, la struttura del brano non è nemmeno aleatoria, poiché non presenterebbe altrimenti certe regolarità (ripetizioni correlate a medio e lungo termine) che diventano riferimenti percettivi essenziali nell'identificazione della forma del brano.

///onepfeig/// a x - (a x^2)
r = 3.862
x = .5

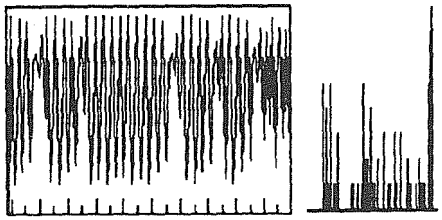
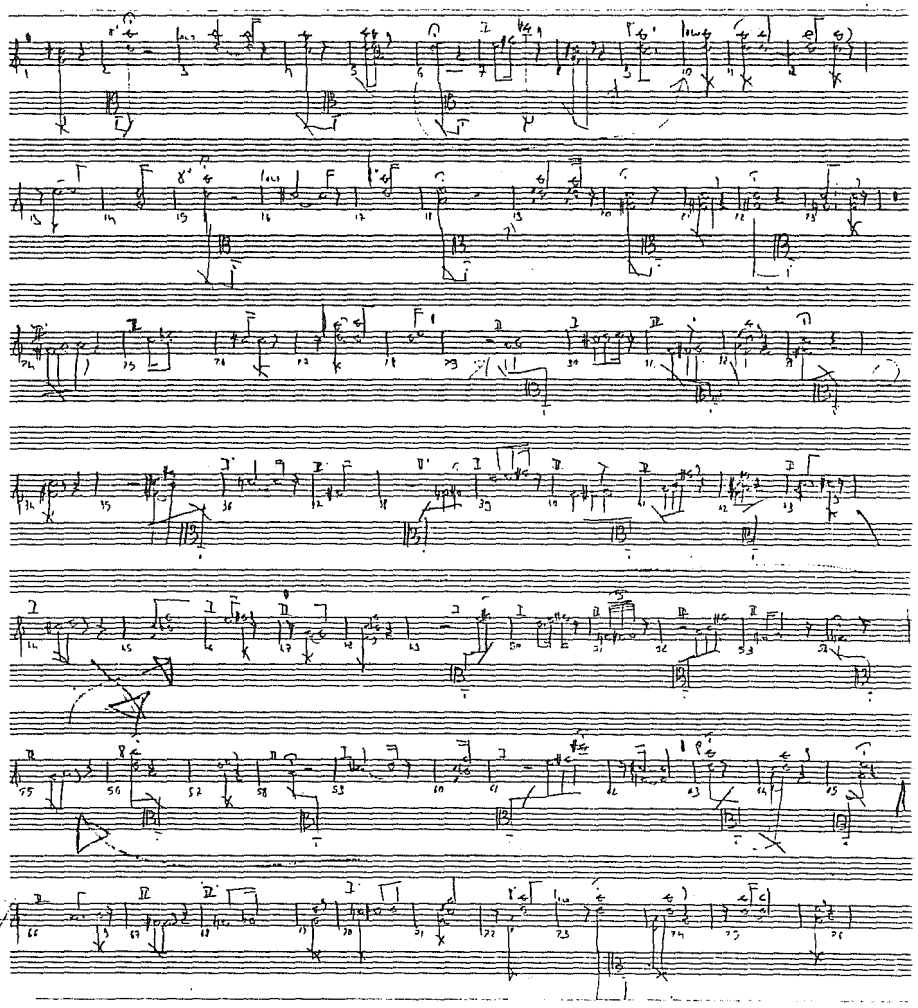


fig.4 - Estratto di *"fractus"*, corrispondente alla prima sezione del brano (sotto), ottenuta riscalando una esplorazione della mappa di biforcazioni per $r = 3.862$ (a fianco) su un insieme di elementi predisposti



3. Procedure di sintesi del suono

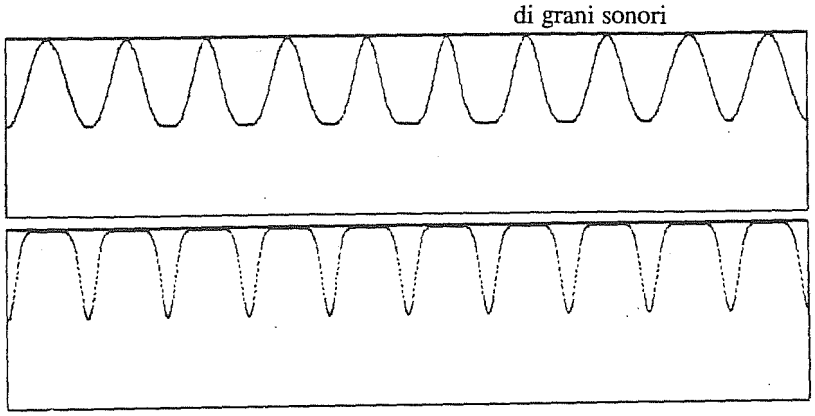
Modelli del caos deterministico come quelli osservati, possono essere utili nel gestire controlli significativi a livello della struttura morfologica di un evento sonoro; nelle principali applicazioni realizzate in tal senso, sono state utilizzate procedure di sintesi granulare. La sintesi granulare generalmente induce a pensare l'organizzazione dell'evento sonoro in uno spazio percettivo continuo che va dalla segregazione di immagini uditive autonome alla fusione di migliaia di dettagli in una singola immagine complessa. La quantità di dati da gestire attraverso elaboratore - i parametri relativi a migliaia di grani sonori - è tale da richiedere approcci di tipo statistico (Roads, 1985; Truax, 1987). Per questo, la necessità di ideare eventi acustici complessi nel dettaglio con mezzi operativi efficaci, ha suggerito l'impiego di sistemi dinamici non-lineari (Di Scipio, 1990; Truax, 1990). La sintesi granulare è legata concettualmente a complesse forme di rappresentazione dei segnali che rivestono attualmente diffuso interesse (cfr. il testo antologico De Poli, Piccialli & Roads, 1991), e l'uso di modelli iterativi sembra coerente al tentativo di ricondurre le strategie organizzative della superficie acustica a forme di auto-organizzazione presenti in fenomeni naturali e sistemi fisici che manifestano sostanziali non-linearità e complessità di comportamento. (Parte delle procedure che verranno ora illustrate, sono state utilizzate in *"ikon"*, per suoni generati mediante elaboratore su nastro a 4 tracce, 1991; altre sono attualmente in fase di studio).

Modelli per la sintesi granulare

Il principio operativo di base non cambia rispetto alle applicazioni macrostrutturali. L'esempio più semplice consiste infatti nel riscalare il valore x_n della n-esima iterazione di un sistema non-lineare entro un dato *range* di frequenza, ed assegnare quindi la frequenza ad un grano di segnale sinusoidale. Le modulazioni di frequenza (e le intermodulazioni di ampiezza dovute alla sovrapposizione di grani simultanei) tracciano allora con notevole precisione l'evoluzione del sistema, caratterizzando lo spettro dinamico risultante. E' utile estendere il controllo all'ampiezza di ogni grano usando, p.e., iterazioni biparametriche oppure mediante iterazioni monoparametriche accoppiate; inoltre è utile rendere la durata di ogni grano funzione inversa della frequenza, accorgimento inerente ad un approccio quantistico alla natura del suono. Una volta stabilito il range di frequenza, occorre sperimentare i vari comportamenti del sistema attraverso il parametro r (e produrre eventuali variazioni con diversi valori in x_0). Nei risultati conseguiti sono uditivamente evidenti le varie transizioni di fase; in generale si può dire che la presenza e l'assenza di attrattori determinano rispettivamente oscillazioni periodiche oppure turbolente ed irregolari. Molte altre combinazioni sono però facilmente sperimentabili, a seguito di punti e cicli instabili o di attrattori strani.

Nella granulazione di suoni già in forma numerica, x_n è riscalato sul numero dei campioni che descrivono il suono originale; il campione così selezionato viene considerato il primo campione del grano n-esimo. Un vettore di riferimento tanto vasto (tra 10^4 e 10^7 campioni, a seconda della frequenza di campionamento e della durata del suono) permette di tracciare l'evoluzione del sistema in modo piuttosto dettagliato. In corrispondenza di punti o cicli limite si avranno leggerissimi spostamenti lungo il file di campioni originale, con conseguenti leggeri mutamenti timbrici ed intermodulazioni tra grani quasi identici, eventualmente sovrapposti in tessiture polifoniche. In corrispondenza di zone caotiche la *texture* sarà più discontinua e confusa, e riorganizza il suono originale in nuove configurazioni di grani sonori. E' utile sottoporre all'azione di filtri passa-basso il segnale in uscita dagli algoritmi di granulazione; infatti nel caso in cui punti fissi e cicli limite assai ridotti vadano a selezionare ripetutamente frammenti di segnale dallo spettro molto largo (per esempio i transienti d'attacco di un suono dall'attacco assai rapido), si avrebbero risultati eccessivamente rumorosi.

fig. 5 - Segnali il cui periodo è usato come inviluppo di grani sonori



L'inviluppo dei grani sonori viene realizzato come un singolo periodo del segnale

$$s(t) = | \sin (2 \pi \omega t)|^p \qquad \text{[eq. 7]}$$

La profondità della curva tracciata dall'ampiezza istantanea di questo segnale è funzione inversa dell'esponente *p*; si è usato anche il segnale complementare

$$s(t) = 1 - | \sin (2\pi \omega t + \pi/4)|^p \qquad \text{[eq. 8]}$$

la profondità della cui curva è invece direttamente proporzionale all'esponente *p*; la forma di questa curva approssima una gaussiana (fig. 5). Il periodo di granulazione è controllato in modo dinamico, generalmente mediante un algoritmo del tipo rumore $1/f^2$ per determinare il valore *p* o il ritardo tra grani successivi (sequenze di campioni nulli).

Un caso particolare, la granulazione di un suono sinusoidale di frequenza crescente in modo continuo da 0 a *Fc/2* Hz, ci riconduce alla sintesi granulare accennata come primo esempio. Tuttavia, se in quel primo esempio lo spettro di frequenza di un grano si riduceva ad una sola componente sinusoidale, qui risulterà leggermente più largo, a causa della modulazione presente nel segnale originale. Ne deriva che identiche successioni x_n possono dar luogo a configurazioni di grani dallo spettro più o meno largo, in funzione della velocità con cui il segnale originale modula in frequenza.

Queste procedure sono attualmente implementate in tempo differito, su un IBM PC 286. In una eventuale implementazione in tempo reale non esisterebbero problemi legati al calcolo delle iterazioni monoparametriche (le cui evoluzioni possono essere anche precalcolate e poste in tabelle). Per la granulazione di suoni reali, invece, un problema attualmente insormontabile sta nella quantità di RAM nella quale il segnale da ridurre in grani è conservato, e che può richiedere molte migliaia di bytes; la

memoria indirizzabile dai DSP in commercio è molto più ridotta, e non rende ottenibili, per ora, i risultati realizzabili in tempo differito.

Dilatazione temporale ed altre soluzioni

Mediante forme di elaborazione granulare del suono, è possibile operare sulla durata del suono senza modificarne il contenuto di frequenza (Arfib, 1990; Jones & Parks, 1988). In termini assai brevi, la dilatazione temporale è ottenuta replicando uno stesso grano sonoro per N volte: poiché lo spettro del grano approssima quello del corrispondente segmento di segnale originale, la ripetizione moltiplica per N la durata del suono originale, senza alterarne lo spettro di frequenza. Procedure dello stesso genere permettono di operare con rapporti di durata non interi, ed inoltre di contrarre la durata del suono. (In una forma analogica non dissimile, questa tecnica fu già utilizzata circa 30 anni fa, al GRM di Parigi, e fu sperimentata in quegli anni anche da K.H. Stockhausen; cfr. Schaeffer, 1966:418-419 e 425-426). Nei suoni così generati risultano spesso chiaramente udibili gli effetti dovuti alla complessa modulazione di ampiezza cui si può ricondurre ogni forma di sintesi granulare. E' possibile evitare questi effetti collaterali, ma talvolta essi possono risultare musicalmente utili, ad esempio per *dotare di altezza* suoni ad altezza indefinita. In effetti, la ripetizione di un grano lungo M campioni realizza una sorta di linea di ritardo con guadagno unitario e ritardo pari a M/Fc sec.; nel suono risultante si avranno quindi risonanze a frequenza $1/(M/Fc)$, $3/(M/Fc)$, $5/(M/Fc)$, ecc..., ed antirisonanze a frequenza $2/(M/Fc)$, $4/(M/Fc)$, ecc... (secondo la tipica risposta in frequenza dei filtri *comb*). Potendo gestire attraverso sistemi non-lineari il ritardo tra campioni (cioè la velocità di granulazione), nonché potendo agire sulla forma dell'inviluppo attraverso il parametro p in eq. 7 e eq. 8, sull'ampiezza del grano e sul numero di grani replicati, è possibile sintetizzare degli eventi sonori che acquisiscono dinamicamente risonanze di guadagno e frequenza variabile, determinate dalle successioni x_n in una o più iterazioni parallele.

Ulteriori "trucchi" consistono in modifiche della fase del segnale nei grani replicati. Truax (1987), p.e., indica di leggere a ritroso la successione di campioni del grano originale: dato l'inviluppo d'ampiezza palindromo e la breve durata, lo spettro di frequenza di un grano non muta significativamente leggendo i campioni per moto retrogrado. E' possibile inoltre invertire il segno del campione, alterando la fase di 180°, ed anche operare spostamenti di fase più raffinati. Quando si alternano grani modificati e grani non modificati, si ottengono delle complicate modulazioni di fase che danno particolare presenza al suono così generato. Infine, effetti molto interessanti sono stati prodotti da granulazioni *a cascata*: suoni già composti da minuscole particelle vengono sottoposti a nuova granulazione, in una sorta di degradazione naturale (o *nebulizzazione*) dell'originale guidata dallo stato evolutivo dei sistemi non-lineari adoperati.

Sviluppi ulteriori

Due ipotesi di lavoro possono essere considerate per ulteriori applicazioni compositive. Nella prima, si tratta di interpretare l'istogramma che cristallizza in una unica immagine l'evoluzione del sistema come una distribuzione di quanti di energia acustica (misurata sull'asse delle ascisse) in una durata assegnata (misurata dall'asse delle ordinate), cioè, in sostanza, come un *pattern* di densità variabile e con proprie curve di ampiezza. Lo spazio delle fasi viene interpretato come *intervallo di tempo*.

Nell'esempio illustrato in fig. 6, viene adoperata la eq. 4, per $x_0 = .124$ e $r = 2.8$; l'evoluzione si assesta presto su un punto fisso. Ad ogni iterazione, x_n è riscalato nell'intervallo $[0,M]$, dove M è un certo numero di campioni nulli, per una durata pari a M/Fc sec. Al campione selezionato viene dato un valore di ampiezza, generando quindi uno *spike* di durata pari a $1/Fc$ sec. (dallo spettro praticamente piatto da 0 Hz a $Fc/2$. Dopo

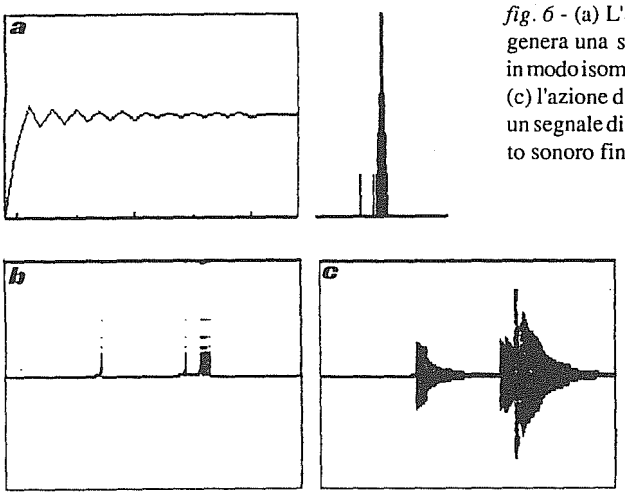
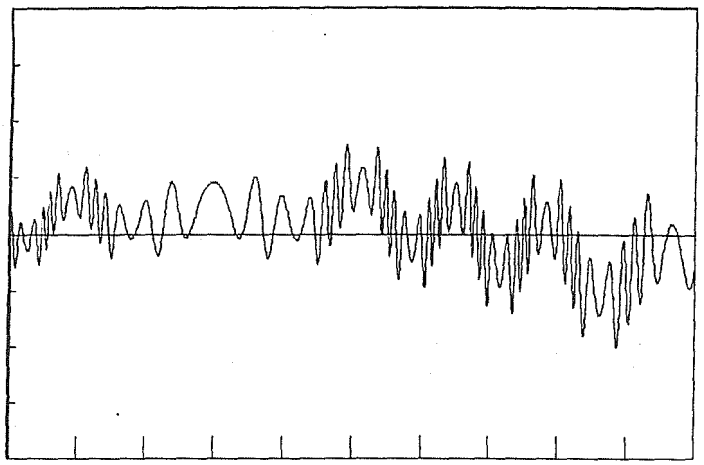


fig. 6 - (a) L'andamento di un sistema non-lineare genera una serie di *spikes* (b) distribuiti nel tempo in modo isomorfo all'istogramma delle sue iterazioni; (c) l'azione di filtri passabanda riduce ogni *spike* ad un segnale di spettro finito, producendo così l'evento sonoro finale

fig. 7 - Un segnale generato mediante la somma dei valori x_4, x_5 ed x_6 rilevati in molteplici iterazioni di uno stesso sistema dinamico non-lineare



molte iterazioni, gli *spikes* saranno concentrati in un gruppo compatto (se l'evoluzione fosse stata caotica, si sarebbe avuta una distribuzione piuttosto uniforme; se vi fosse stato un attrattore strano si sarebbero avuti gruppi di distribuiti in una struttura generale autosimile). La successione di *spikes* così ottenuta viene sottoposta all'azione di filtri passabanda; con filtri del tipo IIR del secondo ordine e per Q molto elevati, ogni *spike* può essere ridotto ad un segnale dalla banda di frequenza molto limitata. La frequenza centrale del passabanda, può essere anch'essa generata da una iteratazione monoparametrica, oppure controllata dinamicamente mediante movimento browniano. Il risultato è un intero macroevento la cui forma temporale di densità/intensità è determinata dallo stato evolutivo della iterazione monoparametrica utilizzata.

I filtri IIR usati in questo esempio sono ottimali per la loro risposta in frequenza, ma presentano una risposta all'impulso troppo rapida ed immediata. Una risposta piuttosto lenta ci riporterebbe ad una forma di sintesi granulare.

L'altra ipotesi, infine, parte dalla constatazione che particolari sistemi non-lineari possono essere utilizzati come degli oscillatori virtuali che determinano la propria forma temporale. Il frammento di segnale in fig. 7 risulta dalla somma dei valori x_4, x_5 ed x_6 in successive inizializzazioni di uno stesso modello iterativo. Senza entrare nel dettaglio, ci limitiamo ad osservare che i risultati sono assimilabili ad una complessa modulazione di fase, le cui evoluzioni spettrali tempo-varianti seguono andamenti curvilinei (peraltro autosimili) percettivamente vicini a quelli rilevabili in alcuni fenomeni naturali - p.e. nelle curve d'ampiezza dei rumori causati dal vento (Carmona & Nualart, 1989).

4. Considerazioni e prospettive

I modelli di sistemi non-lineari permettono di gestire grandi quantità di dati in modo sostanzialmente diverso da quanto è possibile fare con altri mezzi. Nelle applicazioni che fanno uso di procedure di sintesi digitale del suono, la morfologia di interi eventi sonori è catturata nei soli valori di inizializzazione del sistema: il compositore può gestirne perfette riproduzioni, o alterazioni più o meno pronunciate, operando su due sole variabile.

Con l'impiego di simili procedure, l'intervento compositivo anziché assumere un materiale o predeterminarlo, ne provoca una auto-organizzazione a livello microscopico che si traduce in elementi di livello superiore, ovvero in simboli. Si può parlare quindi di approccio compositivo che opera a livello *subsimbolico* (Di Scipio, 1991). Il fatto di adoperare tecniche di *caos deterministico* sia a livello simbolico (come in *"fractus"*) che subsimbolico, ha una sua valenza estetica, che va interpretata in modo adeguato. Piuttosto che unificare la micro e la macrostruttura della composizione (secondo un pensiero musicale ormai storicizzabile), questo approccio tende invece a *confondere l'intervento compositivo tra progettazione dei materiali e loro organizzazione musicale*, rinnovando sostanzialmente il modo di relazione tra compositore e materiale sonoro (Duchez, 1991). Ad esempio, nel progetto di *"ikon"*, ultimato solo di recente, la struttura musicale globale risulta da operazioni di portata locale e circoscritta; essa non è *perseguita* o *raggiunta*, ma cresce assieme al dettagliato e particolare *formarsi* dei materiali stessi. La possibilità di dotare la superficie acustica di adeguata pregnanza percettiva, di renderla traducibile all'ascolto nell'organismo unitario che è detto *forma musicale*, è giocata proprio sulle proprietà di auto-organizzazione evidenziate a più livelli nei modelli presentati, sugli elementi ridondanti ed ordinati che si delineano all'interno di strutture altrimenti caotiche ed imprevedibili.

E' interessante notare che nei processi cognitivi dell'ascolto, esiste una sorta di filtro per il quale solo alcuni dati percettivi partecipano all'integrazione della forma musicale; tale filtro è dato dall'orizzonte temporale entro il quale è possibile la correlazione tra dati percettivi di natura omogenea (McAdams, 1987; 1989). Similmente in fenomeni dinamici esiste un orizzonte temporale entro il quale determinati eventi devono riprodursi perché influiscano su processi a lungo termine, ed oltre il quale quegli eventi (non correlati) diventano invece inessenziali all'evoluzione del sistema. Il senso di un approccio subsimbolico alla composizione mediato da tecniche di caos deterministico, sta allora non solo nel controllare lo stato d'ordine di una struttura (ciò che potrebbe esser fatto anche con altri mezzi), ma soprattutto nel tradurre la dialettica tra determinismo a breve termine e processo indeterminato a lungo termine che è riscon trabile analogamente in forme di auto-organizzazione nel mondo fisico, nei sistemi viventi e nei processi cognitivi dell'esperienza musicale.

References

D. Arfib, "In the intimacy of a sound", **Proc. ICMC-90**, 1990, pp. 43-45

H. Brün, "Infraudibles", in J. Beauchamp & H. Von Foerster, eds. **Music by computers**, New York, J. Wiley & Sons Inc., 1966, pp. 117-121

R.A. Carmona & D. Nualart, **Non linear stochastic integrators, equations and flows**, New York, Gordon & Beach, 1989

P. Collet & J.P. Eckmann, **Iterated maps on the interval as dynamical systems**, Boston, Birkhauser, 1980

B. Degazio, "Musical aspects of fractal geometry", **Proc. ICMC-86**, 1986, pp. 435-441

G. De Poli, A. Piccialli & C. Roads (eds.) , **Representations of musical signals**, Cambridge Mass., The MIT Press, 1991

A.K. Dewdney, "Monti frattali, piante graftali e grafica al calcolatore", in G. Casati ed., **Le leggi del disordine**, Milano, Le Scienze spa, 1991, pp. 117-121

A. Di Scipio, "Composition by exploration of non-linear dynamic systems", **Proc. ICMC-90**, 1990, pp. 324-329

A. Di Scipio, "La musica di due culture: Tracce di una mutazione", in C. Boschi, ed., **Musica/Scienza, il margine sottile**, ISMEZ, Roma, 1991

M.E.Duchez, "L'évolution scientifique de la notion de matériau musical", in J.B. Barrière, ed., **Le timbre, métaphore pour la composition**, Paris, C. Bourgois/IRCAM, 1991, pp. 47-81

J. Gleick, **Chaos**, New York, Viking Penguin, 1987 (trad. it. **Caos, la nascita di una nuova scienza**, Milano, Rizzoli)

D. Jones & T.V. Parks, "Generation and organization of grains for music synthesis", **Computer Music Journal**, vol. 12, n. 2, 1988, pp. 27-34

H. Maturana & R. Varela, **Autopoiesis and cognition. The realization of living**, Dordrecht, D. Reidel Publ., 1980 (trad. it. **Autopoiesi e cognizione. La realizzazione del vivente**, Venezia, Marsilio, 1985)

S. McAdams, "Music: a science of mind?", **Contemporary Music Review**, vol. 2, n. 1, 1987, pp. 1-62

S. McAdams, "Contraintes psychologique sur les dimensions porteuses de forme en musique", in I. Deliège & S. McAdams, eds., **La musique et les sciences cognitives**, Liege, P. Mardaga, 1989, pp. 257-284

J. Pressing, "Nonlinear maps as generators of musical design", **Computer Music Journal**, vol. 12, n. 2, 1988, pp. 35-46

I. Prigogine & I. Stengers, **Entre le temps et l'éternité**, Paris, Arthème Fayard, 1988 (trad. it. **Tra il tempo e l'eternità**, Bollati Boringhieri, 1989)

C. Roads, "Granular synthesis of sound", in C. Roads & J. Strawn, eds., **Foundations of computer music**, Cambridge Mass., The MIT Press, 1985, pp. 145-159

P. Schaeffer, **Traitè des objects musicaux**, Paris, Seuil, 1966

B. Traux, "Real-time granular synthesis with the DMX-1000" (software documentation), Simon Fraser University, 1987

B. Traux, "Chaotic non-linear system and digital synthesis: An exploratory study", **Proc. ICMC-90**, 1990, pp. 100-103

H. von Foerster, "Cibernetica ed epistemologia", in G. Bocchi & M. Ceruti, eds., **La sfida della complessità**, Milano, Feltrinelli, 1985, pp. 112-140

ANALISI PARADIGMATICA DI REPERTORI MONOFONICI

Lelio Camilleri, David Bencini, Michele Ignelzi
Divisione Musicologica del CNUCE/C.N.R.
Conservatorio di Musica L. Cherubini
Piazza delle Belle Arti 2
I-50122 Firenze

E-mail: CONSERVA@IFIIDG.BITNET
Tel. 055-282105
Fax: 055-2396785

Il programma PIAP é un insieme di routines PASCAL per la realizzazione dell'analisi paradigmatica su brani musicali monofonici.

L'analisi paradigmatica é un tipo di strategia analitica che si basa sul concetto dell'unitá significativa e le sue ripetizioni, chiamate sotto-unitá. Come unitá significativa viene definita quella successione di altezze che si ripete almeno due volte esattamente nella stessa forma all'interno del brano. Le sotto-unitá sono invece delle trasformazioni dell'unitá significativa, sia a livello della struttura intervallare che di quella ritmica. Scopo del programma é quello di fornire un facile strumento per realizzare analisi di questo tipo in repertori monofonici, integrato con routines di trascodifica dei dati musicali che traducono dal formato di alcuni codici alfanumerici alla rappresentazione dei dati interna al programma.

Il programma ha un algoritmo che individua l'unitá significativa e la visualizza in fase di output all'utente. In questa fase l'utente puó far eseguire la ricerca delle varie sotto-unitá in due modi: (1) accettando che la ricerca venga fatta con l'unitá

intera, (2) selezionando una qualsiasi parte dell'unitá. Questa possibilitá é stata introdotta per permettere all'utente di fare diverse ricerche sullo stesso brano scegliendo delle parti dell'unitá, in modo da poter poi confrontare e analizzare i vari risultati delle ricerche stesse. Inoltre é in fase di implementazione una funzione in cui l'utente puó specificare di quanto deve essere la deviazione intervallare fra l'unitá e le sue derivazioni. In questo modo l'utente puó facilmente scegliere fra ricerche di derivazioni dell'unitá la cui similaritá sia di maggior grado, e altre in cui vengano inclusi anche segmenti melodici con un minor grado di appartenenza all'unitá prescelta.
Il formato dei dati in uscita é quello del programma SCORE. In questo modo l'utente puó realizzare un numero di files contenenti i risultati delle ricerche e poi visualizzarli con il programma SCORE, ed eventualmente stamparli. Fra l'altro il formato SCORE possiede un tipo di struttura dei dati che permette, eventualmente, anche la memorizzazione dei risultati in un database.
Il programma possiede una propria rappresentazione interna dei dati. Per permettere l'immissione dei dati da altri formati, in modo da poter accedere ad un repertorio di brani piú vasto, sono state create due routine che traducono i file da formato SCORE al formato del programma e ne permettono l'assemblaggio in un unico file. In questo modo l'utente puó tradurre il proprio brano, che normalmente é suddiviso in un certo numero di file SCORE, in un file leggibile dal programma. La possibiltá di usare il formato SCORE da anche l'accesso a vari repertori melodici codificati con

il formato ESAC tramite la routine ESSCORE, scritta da Nigel Nettheim, che é inclusa nell'ambiente di lavoro del programma.

Nella presentazione verrano esemplificate le caratteristiche della procedura analitica implementata, e vari output analitici realizzati dal programma su repertori melodici con cartteristiche distinte.
La presentazione sarbbe integrata da una dimostrazione nello spazio dimostrativo del Colloquio.

Parte 3: Comunicazioni

Capitolo 7: Rapporti di Attività

IRIS: STUDIO REPORT

IRIS

1. INTRODUCTION

IRIS (Istituto di Ricerca per l'Industria dello Spettacolo) is a research organization sponsored by the BONTEMPI-FARFISA Group. Founded in May 1988, under Prof. DI GIUGNO scientific direction and offcially inaugurated on April 26th 1989, during a simposium on music and technology of the '90s. IRIS is now well on its way in its tourth year of existence.

2. OBJECTIVES

The main objective of IRIS was to developed a task force, (with adequate scientific knowledge, technical know-how, and up-to-date facilities) in the field of digital musical instruments. Today IRIS competence extends in numerous disciplines such as:

- Acoustics and psychoacoustics
- Digital Signal Processing, and Algorithms
- Processor architectures
- VLSI (Very Large Scale Integrated circuits) Architecture and Design
- Hardware design and packaging
- Micro-programming, Real-Time Software, Musical and Graphical User Interface (GUI) Software
- and MUSIC!

Main development from IRIS are dedicated Integrated Circuit which enclose in few sq. mm all of IRIS expertise

After IRIS' initial objectives have been reached, the field of endeavour will expand, beyond musical instruments into other related fields (eg. Audio-Visual).

3. ORGANIZATION

As IC' are our main development, IRIS' core is the **Design Center** whose purpose is to turn the engineer's concepts into silicon Chips. The Design Center process involves design and simulations (functional and logic) of the Chip. The Design Center in the interface with the silicon foundry.

Before the silicon step, extensive fundamental research on DSP, Sounds algorithms, processor architectures etc. takes place in the **Research Group**.

As a final step in the process, functional chips are to be tested in operational situations. Developing and building the evaluation and prototype boards in the **Hardware Group**'s task.

Software programming at all levels such as:

- microcode of dedicated DSP's,
- Real time control software hosted by microcontrollers,
- User-friendly graphical interface to configure and debug sound generation boards, is the responsability of the **Software Group**.

4. FACILITIES

IRIS is located 50 km south-east of ROME, 2 km form the "A1 ROMA-NAPOLI" throughway. The 600 sq.mt. center is located in an ancient (but restored!) agricultural building, in the vicinity of an ornithological parc close to the small city of PALIANO.

Computers are IRIS' main tools:

- 5 HP/Apollo Workstations (4000, 5500, 400dl) with Mentor Graphics CAD packages are devoted to the Design Center for VLSI development. Software tools comprise Schematic Editor (NetEd), Logical Simulation (QuickSim), and heuristic Fault analyzer (Quickgrade) with Cells library form various silicon foundries.

- 5 HP/Apollo Workstations (3500, 425t, 400dl), ,5 PC (Compaq and HP) and numerous ATARI of all sizes are the hardware basis of the Software group. C is the main programming language, and experiments on C++are on the way.

- Small listening facilities are found around the building, together with two small studios with analog and digital processing and recording facilities.

- Various peripherals (laser printers, plotter) and analysers (Real-time FFT, Tektronix logical signals analyser) are available. An electronic laboratory in charge of prototype building.

5. COLLABORATION

IRIS's internal industrial activities are mostly devoted to its sponsor, nevertheless the center is also involved in numerous external research programs such as:

- the italian strategic program on "Fast and Parallel Computers", (IRIS is the only industrial research organization of the working group).

- the European ESPRIT II-APBB program, as an active partner of SGS-THOMSON.

- the JESSI program

- ultimately, the ECHOSEA proposal, on a new high-resolution, high acoustic power echographic system for sea bottom exploration was awared the EUREKS status. ECHOSEA, who was initiated by the O.M. Corbino Institute of the NRC (National Research Council) will join seven partners (3 italians, 2 danish, 2 french), and will be coordinated by IRIS, which will develop the DSP part of the project.

Musical cooperation is planned with FRANCE's IRCAM, ITALY's Luciano BERIO, and PADOVA's University.

6. FIRST RESULTS

During last years IRIS has realized several Integrated Circuits for Sound Generation. Most of these will be used in BONTEMPI toys and instruments. Nevertheless IRIS has realized a research circuit (named X20) used as a technological test and workbench.

The X20 circuit is a fully programmable DSP specialized for synthesis and sound transformations. It comes in a 144 pins package and represent roughly 20.000 gates. It is realized in μm CMOS technology.

The X20 can address up to 16 MByte of external memory (RAM or ROM). The X20 is LIW (Long Instruction Word) machine with microprogram instruction range of 512 by 64 bit. Working with a 40 MHz clock frequency, the X20 has a typical 40 kHz sample frequency. Signal Input and Output have a 16 bit precision, but all processing units features 24 bit data path.

A dedicated assembly language (ASM20), has been written (with UNIX's yacc and lex) to allow easy microcode programming.

In order to test extensively the X20, a powerful evaluation board (named SM-1000) has been realized. The board interfaces with an ATARI computer which allow configuration and programming for the board. Digital serial link to specific DAC and ADC boards is provided.

The SM-1000 hardware board features:

- two X-20 processors, for sound generation
- two piggy-board memory board for PCM sound samples (with possible choices from 1 MBytes ROM, 512 kBytes RAM or 64kBytes SRAM (for effects).
- A Motorola 68302 microcontroller, with 1Mbyte RAM and ROM sockets. The MC68302 is SM-1000 board controller.
- Midi Interfaces (In, Out, Thru)
- Parallel Interface to ATARI (8/16 bot wide)
- Clock synchronisation to allow multi board system
- 16 bit digital signal inputs (2), outputs (4)

A proprietary Real Time Kernel (RT20) is embedded onto the MC68302. RT20 achieves the following tasks;

- General scheduling of tasks
- X-20 Loader
- Communications driver (MIDI, ATARI)
- Envelope and LFO (Low Frequency Oscillators) management

The complete sound generation and DAC/ADC boards is housed in a small 19"-2U Rack.

The ultimate programming step is constitued by the user interface program running on the ATARI. Tat program (EDIT20) offers, through an user-friendly graphical interface, several level of programming of SM-1000 board, from the basic function of micro-code definition (with adeguate knowledge of chip architecture and instruction set) to algorithms edition and parametrisation (targeted

to musical applications). What's more, the current version of EDIT20 offers a graphical interactive interface for real-time algorithms design and customisation. Interface with currently available musical softwares (sequencer, samples editors) is also provided, EDIT20 as been used initially for IC and SM-1000 debugging, then for algorithms programmation.

Now, as the complete system (SM-1000 and associated software) is thoroughly programmable, it is used internally at IRIS as a powerful development tool, for sound simulations, sound algorithms experimentations, future VLSI architectures sound evaluation... As a test case, the system has been also used in live musical concert situation.

Studio Report del

Reparto d'Informatica Musicale del CNUCE/CNR di Pisa

via S.Maria 36 - 56126 Pisa
Tel. 050/593276 - Fax 050/576751
E.mail Music5 @ ICNUCEVM.CNUCE.CNR.IT

a cura di

Leonello Tarabella e Graziano Bertini

Abstract

Traditionally Computer Music has been present and active in Pisa for several years. Activities at the Computer Music Department of CNUCE/CNR are carried on by Leonello Tarabella (Coordinator), Graziano Bertini, Alfonso Belfiore and Paolo Carosi. The group mainly deals with the following topics: Personal Computer based Music Workstation design and prototyping, Development of Software for composition and interaction, Education and organization of special events like the recent International Workshop on Man-Machine interaction in live performance.

Keywords: Musical Workstation, Digital Signal Processing, Algorithmic composition, Interaction

Introduzione

Il CNUCE è un Istituto nato nel 1965 come Centro Nazionale Universitario di Calcolo Elettronico sulla base di una convenzione tra IBM ed Università di Pisa per fare da supporto di calcolo all'area cittadina; successivamente (1974) è passato al Consiglio Nazionale delle Ricerche. Nell'Istituto, che conta oggi più di 120 dipendenti di cui la metà circa ricercatori, convivono e si integrano attività di servizio e di ricerca che si espletano nei vari Reparti su settori teorici ed applicativi quali: Linguaggi logici e funzionali, Reti di calcolatori, Controllo volo satelliti, Basi di Dati, Calcolo Parallelo, Ingegneria Strutturale, Telerilevamento, Elaborazione Immagini, Informatica Musicale.

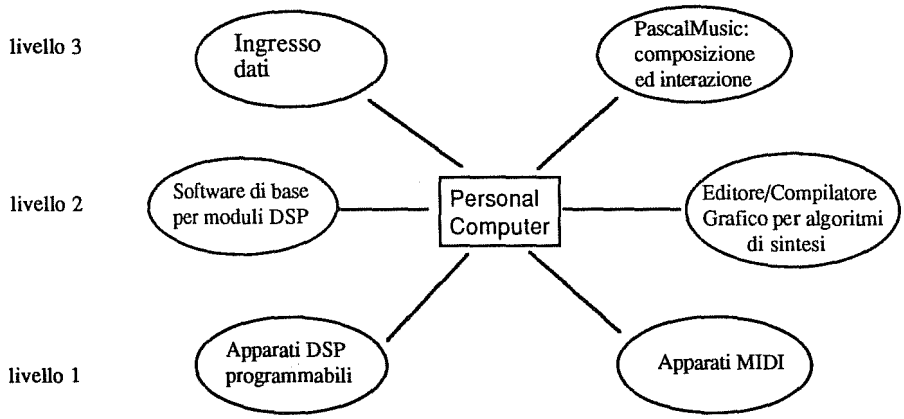
Le ricerche sull'Informatica Musicale vengono svolte in parte in Istituto ed in parte presso il Conservatorio di Musica L.Cherubini di Firenze. Qui, infatti, in virtù di una Convenzione tra l'Istituto CNUCE e il Conservatorio stesso, vengono svolti studi su argomenti di analisi musicologica, sistemi esperti e musicologia computazionalepresso ad opera del M° Lelio Camilleri (Collaboratore Esterno del CNUCE), del Dr. Francesco Carreras, ricercatore del CNUCE, e di altri collaboratori [vedi Notiziario AIMI n.55, Marzo 1990].

A Pisa, presso l'Istituto CNUCE le attività vengono condotte all'interno del Reparto di Informatica Musicale da Leonello Tarabella (coordinatore), Graziano Bertini, Alfonso Belfiore e Paolo Carosi, e riguardano principalmente la Ricerca, la Didattica, la Promozione e la Produzione musicale. La ricerca è prevalentemente di tipo applicativo e rivolta alla realizzazione di supporti hard e soft da utilizzarsi in stazioni di lavoro musicale. Vengono curati i seguenti temi: - progettazione di sistemi digitali per la sintesi e l'elaborazione di segnali audio; - progettazione e realizzazione di strumenti software per la definizione ed il controllo di algoritmi di sintesi e di trattamento di segnali audio; - sviluppo e formalizzazione di metodologie compositive con particolare riferimento a tecniche di interazione e live-performance; - progettazione e realizzazione di interfacce uomo-macchina basate su sensori speciali.

Stazioni di lavoro per segnali acustico/musicali

La disponibilità di Personal Computer e componenti elettronici ad altissima integrazione con prestazioni sempre crescenti, permette oggi la progettazione di apparati con varie funzioni complesse, utilizzabili per la realizzazione di stazioni di lavoro musicali riconfigurabili al fine di consentire un uso differenziato adattabile a diverse situazioni d'impiego. Dal momento che le apparecchiature in commercio come sintetizzatori, Campionatori, Processori di suono, etc. non soddisfano completamente le richieste nei settori più avanzati della ricerca sul suono in sé e della composizione musicale vera e propria, è sentita l'esigenza di avere a disposizione macchine flessibili che consentano l'effettiva programmabilità di algoritmi di sintesi, e di elaborazione estemporanea di segnali reali.

Facendo riferimento ad uno schema di principio di stazione lavoro per elaborazione di segnali acustico/musicali come quello della seguente figura dove sono riportate alcune funzioni essenziali, è possibile individuare più livelli di intervento: hardware, software di base, software per la composizione e l'interazione. [1] [2]



Hardware

Per quello che riguarda i dispositivi hardware, il nostro intervento consiste principalmente nel progetto e realizzazione di moduli che utilizzano dispositivi VLSI come microprocessori specializzati per il Digital Signal Processing, circuiteria PAL (Programmable Logic Array), Dual port RAM. Per le apparecchiature Midi commerciali ci si limita a realizzare le opportune interfacce di controllo, oltre che ad utilizzarle in misura della loro gamma di prestazioni. Dopo un primo periodo di realizzazioni prototipali con microprocessori della famiglia DSP della Texas Instruments [3], curata presso i laboratori di progettazione CAD e montaggio dell'Istituto di Elaborazione dell'Informazione (I.E.I./CNR) di Pisa, sono state recentemente sviluppate alcune versioni di schede per Bus Pc-IBM compatibile, disegnate intorno al TMS320C25, in collaborazione con il Centro Tecnologico Culturale Leonardo di Massa, che ne cura peraltro la commercializzazione con la sigla di **Leonard 'C-25** [4]. Caratteristiche principali di tali schede sono la presenza di un'elevata quantità di memoria a zero wait-states (128K-words DataRAM, 64K-words ProgramRAM, 4K-words Eprom), e di porte seriali e parallele di comunicazione con dispositivi di conversione DA e AD.

Dal momento che in generale una singola scheda non soddisfa tutte le esigenze che si incontrano in composizioni musicali complesse, viene portato avanti il progetto di un apparato *multi-processor* con architettura più adatta a questo tipo applicazioni, [5] le cui caratteristiche di modularità ed espandibilità consentono capacità di elaborazione dell'ordine dei 100 MIPS. Al momento è in corso di realizzazione un primo sistema che utilizza più esemplari di schede Leonard C-25 poste sul bus di calcolatore della classe PC-IBM, e che consentirà di sviluppare e testare il relativo software.

Software di base

Per quello che riguarda il software di base per il controllo del sistema multi-processor, va detto che è stato progettato un particolare programma le cui caratteristiche possono classificarlo come Sistema Operativo, le cui funzionalità principali sono quelle di: - distribuire in maniera il più possibile trasparente all'utente le risorse computazionali disponibili complessive nella fase di configurazione iniziale del sistema (questo problema di tipo generale [6], comporta la messa a punto di criteri di scelta per l'*allocazione* degli algoritmi di sintesi e di elaborazione sui diversi moduli); - la gestione run-time dei parametri di controllo per l'esecuzione musicale.

E' necessario puntualizzare che per *algoritmi di elaborazione* si intende sia la manipolazione e trattamento di segnali audio di tipo classico (effettistica), sia il complesso di operazioni di tipo analitico come l'estrazione di valori caratteristici (pitch, volume medio...) da utilizzare in tempo reale.

In questa categoria di tools di base può essere inserito anche l'Editore/Compilatore Grafico realizzato su macchine Macintosh, che consente di progettare in forma grafica schemi di algoritmi di sintesi sonora e di filtraggio digitale: lo strumento mette a disposizione un piano di lavoro ed una palette di icone che rappresentano simboli tipici di assemblaggio di algoritmi di sintesi come: generatori, operatori aritmetici, linee di ritardo, generatori di inviluppo, etc. Caratteristiche principali di questo strumento sono la possibilità di utilizzare in uno schema altri schemi più complessi predefiniti (macro) e la verifica della correttezza sintattica e semantica durante la fase di disegno; la fase finale prevede la traduzione dello schema disegnato in linguaggio macchina di un generico DSP [5].

Software per la composizione e l'interazione

Seguendo la tradizione che riconosce ai linguaggi di programmazione le caratteristiche di strumento di per sé sofisticato e ricco di potenzialità espressive per il controllo dei parametri acustico/musicali, è stata sviluppato un particolare ambiente compositivo che consiste nell'uso di un linguaggio standard opportunamente arricchito di routines dette *primitive del nucleo*. Il linguaggio scelto è il Pascal, e lo strumento realizzato è il PascalMusic [7]

Seguendo questa filosofia di lavoro, un brano musicale viene dunque descritto in forma di programma: al momento dell'attivazione, il "programma/brano" opera una sorta di *meta-compilazione* della successione degli eventi sonori relativi al brano stesso generando un insieme di strutture dati lineari. Per ottenere l'esecuzione musicale vera e propria è necessario che nel programma stesso sia presente, come ultima istruzione, la chiamata alla routine del nucleo *Execute* che, a partire dalle strutture dati generate nella fase precedente, produce in un secondo momento il codice Midi (o Midi-like) che alimenta le apparecchiature di sintesi.

Il fatto di non generare direttamente nella prima fase il codice Midi, è determinato dalla necessità di poter interagire dall'esterno e di influenzare la composizione al momento dell'esecuzione. A questo scopo, del nucleo fa parte un sottoinsieme di primitive del tipo *Select* e *Loop* con le quali è possibile, durante la prima fase, sofisticare a piacimento la linearità della successione degli eventi con la generazione di punti di diramazione e/o di ciclo: nella seconda fase, quando la *Execute* incontra questi punti speciali, prende decisione sul da farsi in base ad informazioni provenienti dall'esterno (MidiIN). Differenti sezioni di una composizione possono così presentare situazioni di completo automatismo, di totale controllo dall'esterno, oppure un uso misto delle due modalità. E' infatti possibile interagire con la composizione (o meglio, con il programma che la definisce) per mezzo di dispositivi di input gestuali come tastiere dello standard Midi e più in generale di *sensori di movimento* da parte degli esecutori .

Il PascalMusic è stata implementato su macchine MacIntosh e PC-IBM (e compatibili), e utilizzato con efficienza e successo in varie occasioni seminariali dimostrative, compositive e concertistiche. E' al momento in fase di studio e di realizzazione una seconda versione di PascalMusic che fa uso di tecniche di programmazione concorrente: in questo modo è l'effettiva esecuzione parallela di più parti dello stesso programma che realizzano l'esecuzione musicale, e l'interazione viene realizzata agendo direttamente sulle strutture dati e di controllo, sulla base di valori parametrici provenienti dall'esterno.

Dispositivi di ingresso dati

Nelle esecuzioni musicali che prevedono situazioni interattive e di live-performance, vengono utilizzati dispositivi di input reperibili in commercio come le tastiere Midi, drum-pads etc.. In queste sistuazioni vengono inoltre utilizzati speciali sensori di movimento che sono progettati e realizzati in collaborazione con l'ARTS (Advanced Robotics Technology and System) Laboratory della Scuola di Studi Superiori Universitari e di Perfezionamento S.Anna di Pisa, come ad esempio il *Pyramid Infrared Midi Event Generator* e la *Light Baton.* [8,9]

Didattica e manifestazioni

In questi ultimi anni sono stati svolti numerosi seminari e corsi come il Corso sperimentale di Informatica Musicale tenuto presso il Conservatorio L.Boccherini di Lucca e presso la Scuola Musicale di Pisa. Viene inoltre svolto annualmente presso il Dipartimento di Informatica Università di Pisa, un corso integrativo di Informatica Musicale. In questa attività vanno infine incluse le tesi di laurea di studenti di Informatica e di Ingegneria Elettronica. Vengono periodicamente organizzati seminari, manifestazioni e concerti di Computer Music. Di rilievo, il recente *Workshop sull'Interazione Uomo-Macchina nella Performance dal Vivo* svoltasi presso la Scuola S.Anna di Pisa, che ha visto la presenza di numerosi esponenti della Computer Music nazionale ed internazionale, e l'invito all'Accademia Chigiana di Siena nell'ambito del Congresso Annuale A.I.C.A 1991.

Leonello Tarabella si è laureato in Informatica a Pisa, ha studiato musica e tecnica strumentale del sassofono, ed ha trascorso periodi di studio presso l'EMS del MIT di Boston ed il CCRMA della Stanford University. Dal '77 è al CNUCE di Pisa, prima come borsista e poi come Ricercatore, nel Reparto di Informatica Musicale. Tiene un corso di "Elaborazione di segnali acustico/musicali" presso il Dipartimento di Informatica dell'Università di Pisa.

Graziano Bertini si è laureato in Informatica a Pisa. Dal '75 è ricercatore presso il Reparto di Elaborazione Segnali ed Immagini dell'Istituto di Elaborazione dell'Informazione (IEI/CNR). E' stato responsabile dell'equipe che progettò e realizzò il sintetizzatore audio TAU2, ed è tuttora impegnato nelle attività condotte presso il Reparto di Informatica Musicale del CNUCE.

Alfonso Belfiore si è diplomato in pianoforte ed in composizione presso il Conservatorio G.Verdi di Milano; ha studiato Direzione di Orchestra e composizione organistica. Dal 1980 è titolare della Cattedra di Composizione Musicale Elettronica del Conservatorio di Padova. E' Collaboratore esterno presso il Reparto di Informatica Musicale del CNUCE.

Paolo Carosi si è diplomato in flauto traverso presso l'Istituto Musicale L.Boccherini di Lucca; ha frequentato i corsi di perfezionamento dell'Accademia Chigiana; ha superato l'esame di contrappunto e fuga. Ha studiato elettronica ed informatica ed ha frequentato la Scuola di Musica di Fiesole e lo Studio di Fonologia Musicale del Conservatorio di Firenze. E' Collaboratore esterno presso il Reparto di Informatica Musicale del CNUCE.

Riferimenti

[1] Bertini G., 1990. *Giornata di Lavoro (25/9/90) Progetto Finalizzato,* Sistemi Informatici e Calcolo Parallelo, Sottoprogetto 2: Processori dedicati.

[2] Tarabella L., 1987. *The Primula Machine* - Computer Music Journal (MIT Press) - 11(2)

[3] Tarabella L., Bertini G., 1987. *Un sistema di sintesi ad elevate prestazioni controllato da personal computer* - Quaderni di Musica/Realtà n. 14: Atti VI CIM - Unicopli, Milano

[4] Bertini G, DelDuca L., Marani M., 1991. *The Leanard 'C25 system for real-time digital signal processing (abstract)* - International Workshop on Man-Machine in Live-Performance - Pisa

[5] Bertini G.,Tarabella L., 1989. *A digital signal processing system and a graphic editor for synthesis algorithms*, Proceedings of the ICMC89 - CMA, POBox 1634, S.Francisco, Ca, USA.

[6] David J., 1990. *Efficient Dynamic Resource Menagement on Multiple DSPs, as Implemented in the NeXT Music Kit* - Proceedings of the ICMC90-CMA, Glasgow.

[7] Tarabella L., 1990. *Algorithmic composition and interaction*, Array CMA Newsletters, 10(4)

[8] Buttazzo et alii, 1991 - *Infrared-Based MIDI event generator* (abstract, Atti in stampa) - International Workshop on Man-Machine in Live-Performance - Pisa

[9] Carosi P.,Bertini G., 1991 - *Light Baton: a system for conducting computer music* (abstract) - International Workshop on Man-Machine in Live-Performance - Pisa

CRM: FROM THE FLY 10 TO THE FLY 30 SYSTEM

A. De Vitis*, M. Lupone°, A. Pellecchia^

* In charge of hardware at the CRM - Centro Ricerche Musicali
Via Lamarmora 18 - I - 00185 Roma
Tel: 39-6-4464161 Fax: 39-6-7312195
Cod. Anagrafe Ricerche: 707910SU

° Art Director at the CRM - Centro Ricerche Musicali - Roma

^ Scientific Director at the CRM - Centro Ricerche Musicali - Roma

Keywords: Real-Time Sound Analysis/Synthesis, DSP, Composition, Musicology, Acoustics, Psychoacoustics, Floating Point Calculus, User-Friendly Interactive Software, Self-Learning promoting System.

1. Introduction

Over the last few years, the level of development in technology and informatics in the area of numerical processing of sound signals has rapidly made the latter a valuable instrument for all those disciplines, artistic and scientific, which link their method and results to the creation, analysis, synthesis and processing of sound. Because of this, vast sections of *contemporary music*, *musicology*, *ethnomusicology* and *sound restoration* on the one hand, and *acoustic physics*, *psychology of perception*, *sonology* and *organology* on the other, are discovering not only techniques and innovations for their own applications, but above all, they are experimenting with the possibility of creating spheres of interdisciplinary (or simply multimedial) research. By definition, they measure themselves against the very real opportunity of reuniting humanistic with scientific thought, the culture of the non-measurable with the experimental tradition.

In this context, in the spring of 1990 and on the initiative of a group of CRM researchers, the research project called *Fly 30* was begun. The different cultural and professional backgrounds of the members of the team (musicians, physicists, engineers and musicologists) united by a common appreciation of interdisciplinary problems and all specialising in informatic systems, produced the group's present capacity to create a digital system oriented towards the synthesis, analysis and real-time processing of sound signals. This system develops an immediate interaction with the user and is flexible and adaptable to the different scientific and artistic needs.

2. From the Fly 10 System to the Fly 30 System

The *Fly 30* system derives from the *Fly 10* system projected in 1984 by Michelangelo Lupone (first version for APPLE 1984, second version for IBM PC 1986) and is based on the high-speed processor TMS 32010.

The success that *Fly 10* has achieved in this period, in both the artistic and scientific fields, is due to the design philosophy which, by maintaining high calculating power, has also enhanced the user-interaction, facilitating all input functions and data control. The Fly system, originally planned to assist in the work of musical composition by real-time sound synthesis, has shown a much wider functional application based on flexible algorithms and general concepts such as Frequency Modulation, Additive Synthesis, Amplitude Modulation, etc.

The total programmability of the system, and therefore its adaptability to specific needs in different areas of research, has enabled it to be utilised in various fields:

- psychoacoustics: the FIAT Research Centre (C.R.F.) has tested short-burst easily identifiable semantic sound signals for use in the car industry;

- organology: instrument timbres and control algorithms have been tested and perfected, particularly as to what concerns industries of musical instruments;

- composition: since 1985, 30 works have been completed, five of which are suitable for real-time performance by a pianist.

Another important result of the research developed by CRM using *Fly 10* was the project *Musica Infinita*, commissioned in 1986 by ISMEZ (Institute for Musical Development in Southern Italy) and completed the following year.

The project, based on several studies of AI (artificial intelligence), was able to produce musical forms by computer which had the capacity for autogeneration and evolution, while keeping unchanged the rigorous structures (multiple, double, triple, reversible canons) of the counterpoint related to the armonic-tonal system.

The *Fly 30* project, based on preceding experiences such as these and particularly on the knowledge acquired through *Fly 10*, is aimed specifically at rethinking traditional algorithms of numerical calculus in order to achieve their implementation on a new high-speed precision processor (see following pages). It also aims at taking up the "challenge" posed by the latest generation of DSP processors, testing both original calculus algorithms and original real-time control systems of relative parameters, with the ultimate aim of achieving a **dynamic processing of sound signals**.

The digital system called *Fly 30* is therefore a hardware/software system with the following principal characteristics:

1 - capacity to analyse, synthetise and process real-time sound signals

2 - user-friendly interactive software complete with advanced graphic editor, examples and user's manual

3 - easy connection to a personal system (requires an IBM compatible PC-AT with a working DOS system)

4 - modularity (we predict 4 configurations to run from 1 to 4 Sonitech *Spirit 30* boards)

5 - very high precision and velocity computation (data of 32 bits in floating point, parallel processes such as multiplication and addition in 60 nanoseconds).

These characteristics are made possible by the use of the numerical superprocessor TMS 320C30 which is the heart of the *Fly 30* system. It is installed on a program connected directly to the bus of the PC-AT which gives high-speed communication and therefore the exchange in real-time control parameters set by the user.

The *Fly 30* project aims at maximising the interaction and the interdisciplinary relations on which research in the above-mentioned areas is based.

3. Hardware/Software System

The *Fly 30* system consists basically of the following hardware subsystems (fig. 1):

1 - IBM-AT or compatible Personal Computer, functioning as the host computer for the system, complete with colour monitor, a 640Kbytes memory (minimum capacity) and a 40 Megabytes hard disk.

2 - Sonitech *Spirit 30* Board, functioning as the numerical signal processor, based on the DSP TMS320C30, complete with interface logic with the host computer, serial interface for other programs and a 160x32 bits memory.

3 - Sonitech *IC 100* Conversion Board, interfacing with electronic analog devices, based on a 16 bits ADC with a maximum sampling frequency of 100 KHz, a 16 bits DAC and a serial interface logic with the *Spirit 30* board.

The system offers the following resources to users (fig. 2):

1 - Graphic Editor with "sound in line" for the creation of instruments, orchestras and forms of interactive processing.

2 - *Fly 30* Compiler for the creation of complex structures and the linkage of procedures.

3 - Analysis and debug modules of the *Fly 30* system.

3.1 Sonitech *Spirit 30* signal processing board

The *Spirit 30* processing board (fig. 3) has 160x32 bits of memory and is equipped with an interface logic for the AT computer and other similar programs. Stored in the board memory are the programs to be carried out by the TMS320C30 numerical processor, and the numerical data which carry the digitalised information of the signal.

The interface with the computer makes for a rapid exchange of information between the *Spirit 30* board and the computer itself. This allows to achieve a real time system with the possibility of modifying the signal processing while it is occurring. Further, the guest computer can check the work of the board and present in a graphic form what is being processed.

The serial interface allows parallel connection with other *Spirit 30* boards in such a way as to obtain a power system equal to the number of boards.

3.2 Numerical Processor TMS320C30

This processor was projected and built by TEXAS INSTRUMENTS especially for the work of numerical signal processing. It is the most advanced model of the whole family of TMS320xxx processors and is amongst the most sophisticated numerical processors on the market. It can carry out operations on 32 bits of data in 60 nsecs, and has an internal hardware design capable of carrying out more parallel operations and therefore of reaching a greater computing speed. The TMS320C30 processor is capable, for example, of simultaneously carrying out multiplication on different data in the same time cycle.

The assembly instructions permit the carrying out of an FFT (signal transformation in the frequency domain) with excellent results.

A specialized set of instructions allows for optimal layout of algorithms of a numerical filtering of the signals.

The TMS320C30 is internally equipped with the interfaces necessary for communication with the outside world. It has a programmable DMA for the fast exchange of data between the memories and the peripherials. It is equipped with a serial interface programmable for the exchange of information with other TMS320C30 processors or with other systems.

It is equipped with a programmable timer as well as a multifunctional clock to obtain the most complete set of synchronised sounds. It has 2 methods of access: one primary and one secondary, capable of moving as much as 32 bits of information. It has a signal system which allows either multiprogramming or synchronisation of parallel processes stored in different processors. It can address memories of up to 16x32 Mbits, divisible into one or more banks as desirable.

3.3 Sonitech *IC 100* Conversion Board (fig. 4)

The two systems DAC (Digital to Analog Converter) and ADC (Analog to Digital Converter) are used in the *Fly 30* system as interfaces to the real world of sound signals and are implemented in the *IC 100* conversion program. This allows an A/D and D/A conversion at 16 bits of information with a sampling frequency of up to 100 KHz.

The components used and the features implemented provide maximum reliability and a signal/noise ratio of over 90dB.

References

ARFIB, D. "Digital synthesis of complex spectra by means of multiplication of non-linear distorted sine waves" **Journal of the Audio Engineering Society** (1979)

ASTA, V. / SCHAUVEAU, A. / DI GIUGNO, G. / KOTT, J. **Il sistema di sintesi digitale in tempo reale 4x Automazione e Strumentazione,** vol. 28 (1981)

DE VITIS, A. "Introduction on Fly 30 Project: a system for real time sound analysis and synthesis" **proceedings of International WORKSHOP on Man-Machine interaction in live Musical Performance** (Pisa, 1991)

GREY, J. M. / MOORER, J. A. "Perceptual evaluation of synthesized musical instrument tones" **Journal of the Acoustical Society of America** (1977)

LUPONE, M. "System Fly" **VI Colloquio di Informatica Musicale, Quaderni di Musica e Realtà** (1985)

LUPONE, M. "La musica elettronica" - intervista - **Professione Musica** (1988)

MCADAMS, S. / BREGMAN, A. "Hearing musical streams" **Computer Music Journal** (1979)

MOORER, G. A."Signal processing aspects of computer music - A survey" **proceedings of the IEEE** (1977)

PELLECCHIA, A. "Real time DSP system Fly 30: scientific research and musical performance" **proceedings of International WORKSHOP on Man-Machine interaction in live Musical Performance** (Pisa, 1991)

OPPENHEIM, A. V. / WEINSTEIN, C. J. "Effects of finite register length in digital filtering and the fast Fourier transform" **Proceedings of the IEEE** (1972)

OPPENHEIM, A. V. / SCHAFER, R. W. **Digital signal processing** (Prentice-Hall 1975)

SCHROEDER, M. R. "Natural sounding artificial reverberation" **Journal of the Audio Engineering Society** (1962)

SCHROEDER, M. R. "Digital simulation of sound transmission in reverberal space, part 1" **Journal of the Acoustical Society of America** (1970)

SONITECH, Int. Inc. **Spirit 30 system: technical reference manual - version 2.0** (1989/90)

SONITECH, Int. Inc. **Spirit 30 system: IC-100 user manual - version preliminary** (1989/90)

TEXAS INSTR., **TMS320C3x user's guide** (1989)

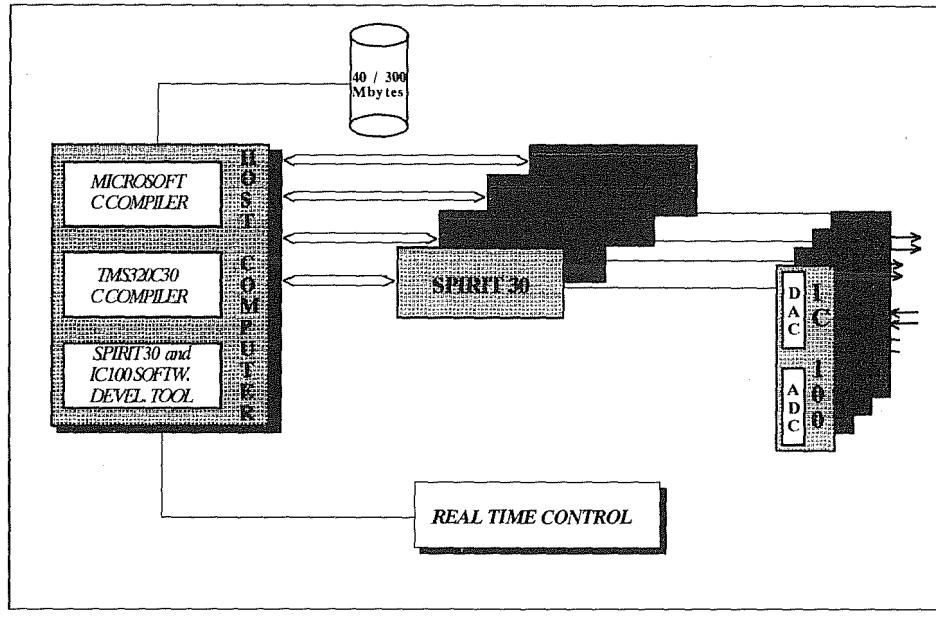


FIG. 1 - Fly 30 System

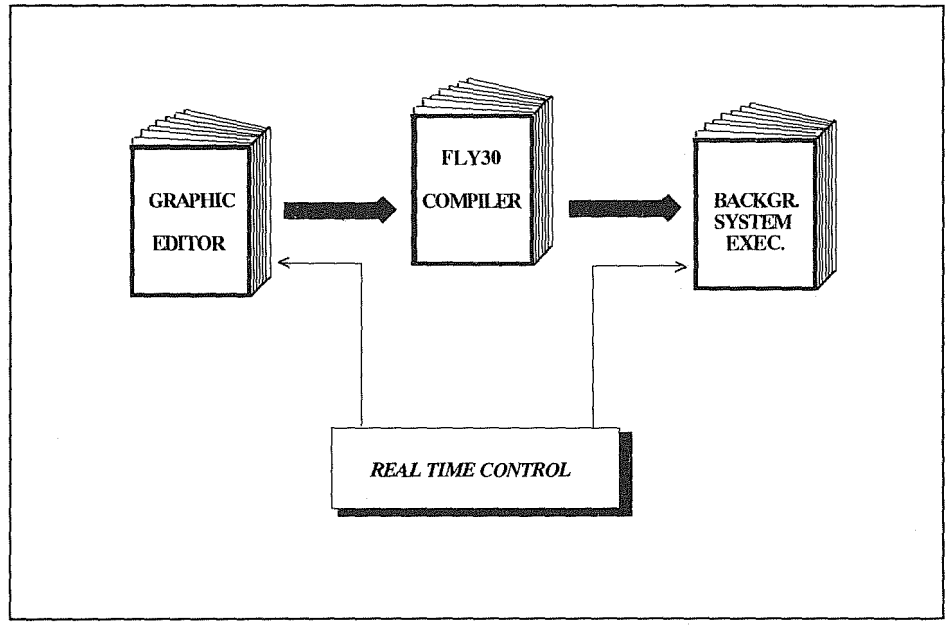


FIG. 2 - Fly 30 software system

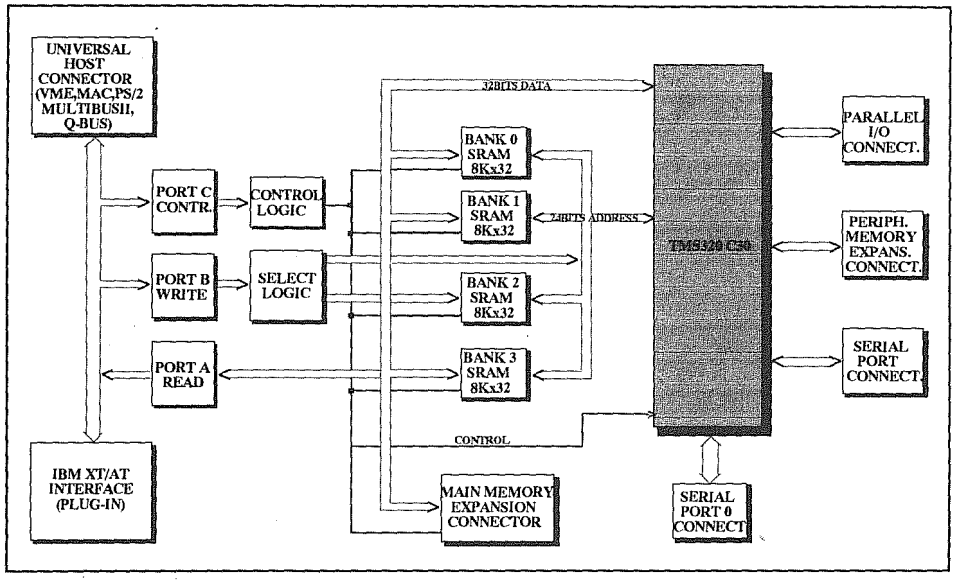


FIG. 3 - Spirit 30 architecture

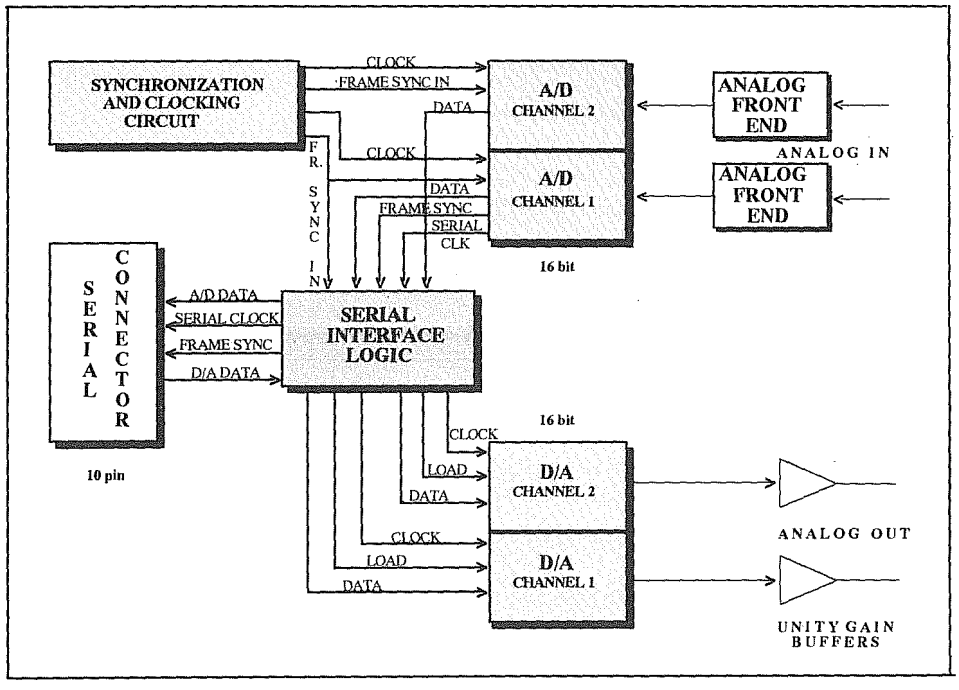


FIG. 4 - IC-100 block diagram

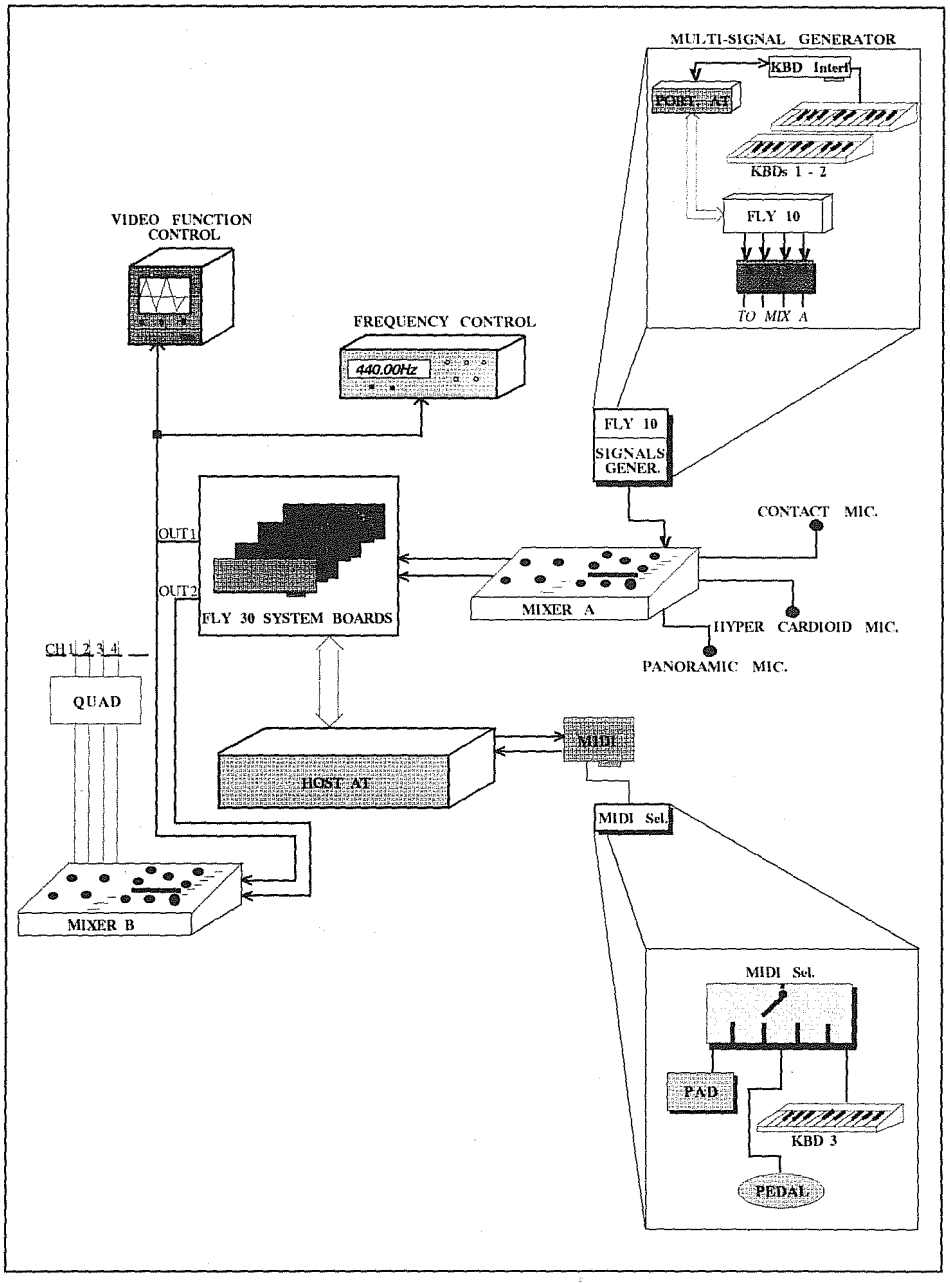


FIG. 5 - Fly 30 System: a CRM complete workstation

Capitolo 8: Dimostrazioni, Video, Poster

INTERAZIONE DI COMPUTER MUSIC E COMPUTER GRAPHIC IN ESECUZIONI DAL VIVO

(Computer Music & Graphic Interaction in Live Performance)

Mauro Graziani

C.S.C.: Centro di Sonologia Computazionale
Università di Padova, Via S.Francesco 11, 35121 Padova, Italy
E-mail: music01@unipad.cineca.it
Tel. (049) 8283711 - Fax 8754094

Keywords: MIDI, live performance, multimedia

ABSTRACT

L'autore presenta i frutti della propria ricerca finalizzata alla definizione delle modalità di interazione audio-video e alla elaborazione di ambienti esecutivi che permettono al musicista di controllare simultaneamente e in tempo reale la parte grafica e quella musicale in esecuzioni dal vivo.

Introduzione

L'uso dell'elaboratore sia nella produzione di immagini che nella creazione di eventi musicali ha prodotto una reale svolta nello studio dei rapporti fra immagine e suono. Per la prima volta un solo mezzo di produzione unifica i due mondi dando la possibilità di sperimentare l'applicazione delle idee su entrambi i media.
Soprattutto, però, il computer ha dato la possibilità di sperimentare l'impiego di algoritmi identici in entrambi i campi, alla ricerca di un parallelismo a lungo sognato. Come si sa, a livello di elaboratore, sia le immagini che i suoni sono il risultato di calcoli i cui dati di uscita possono essere indifferentemente utilizzati sia per accendere dei pixels che come campioni sonori.

A mio avviso, comunque, la cosa più interessante è la possibilità di costruire parti sonore e grafiche come proiezioni della stessa organizzazione formale, metodo da me utilizzato, in modo più o meno spinto, nella realizzazione del software per la serie Mappe di Luce e Suono che si compone sia di installazioni audio-visuali che di pezzi eseguiti dal vivo dall'autore e presentati in varie occasioni, fra cui la Biennale di Venezia 1986.

L'AMBIENTE ESECUTIVO

Questi lavori sono organizzati come ambienti esecutivi nei quali alla tradizionale parte sonora è affiancata una parte grafica generata, in tempo reale, da un personal computer che entra a far parte della rete midi e riceve comandi esattamente come il sintetizzatore del suono. Accanto ai tradizionali comandi midi per la gestione del suono viene utilizzata una serie di comandi grafici. A partire da queste primitive, vengono costruite delle strutture in cui vari comandi grafici e sonori vengono associati.
Ogni comando inviato dall'esecutore tramite la tastiera o il mouse ha, quindi, due aspetti: uno grafico, l'altro sonoro. La preminenza dell'uno o dell'altro è stabilita all'atto della progettazione dell'ambiente esecutivo e può anche cambiare nel corso del brano.
L'autore intende presentare la struttura di alcuni ambienti esecutivi audio-video da lui stesso elaborati, nonché le modalità di interazione che possono essere attivate nel corso dell'esecuzione.

UNA WORKSTATION PER L'ELABORAZIONE NUMERICA DEI SEGNALI MUSICALI E DELLA VOCE

A.Piccialli, P.Basile, M.Sala, P.Simone, S.Vergara

Dipartimento di Scienze Fisiche Universita' di Napoli

ABSTRACT

This paper describes a modular Digital Workstation for musical and vocal signal processing. Modular structure allows to increase Workstation's potentialities by adding new Modules. Final aim is to make the applications of digital processing techniques to musical and vocal sound easy and affordable. The paper focuses on the description of the Modules actually available and on the project of a synthesis Module for vocal sounds. The software has been implemented in TurboPascal and C, running on MS-Dos systems.

1. Introduzione

L'applicazione delle tecniche di Digital Signal Processing ai suoni musicali e vocali dischiude una vasta gamma di possibilita' per lo studio di tali suoni, consolidando risultati già noti e, nello stesso tempo, stimolando nuove prospettive di ricerca. Le tecniche di analisi-sintesi, generalmente basate sulla considerazione di un modello del segnale da elaborare, si rivelano prodighe di sviluppi dal punto di vista della classificazione e della ricerca acustico-musicale sul suono. Tecniche concettualmente simili ma, di fatto, diverse negli obiettivi, e che possono essere denominate di codifica-decodifica, sono di notevole interesse dal punto di vista della compressione dell'informazione. Sono disponibili, inoltre, numerose classi di tecniche, applicate con buoni risultati ai segnali vocali, per le quali lo studio di possibili applicazioni ai segnali musicali costituisce, di per se', uno stimolante argomento di ricerca. Le trasformazioni omomorfe e le tecniche di predizione lineare, ad esempio, presentano potenzialita'

che senz'altro giustificano uno studio intensivo sulle loro possibilita' di applicazione ai segnali musicali.

Il progetto della Workstation che descriviamo e' nato dall'esigenza di raccogliere, in un pacchetto software versatile, un insieme di strumenti particolarmente adatti all'elaborazione dei segnali musicali e vocali. L'obiettivo e' quello di poter disporre, accanto a tecniche tradizionali ben collaudate, di tecniche di piu' recente concezione, con la possibilita' di mediare il punto di vista delle prime con le potenzialita' offerte dalle seconde. La realizzazione della Workstation e' basata su una struttura modulare che rende possibile ogni ampliamento semplicemente aggiungendo nuovi moduli.

2. Descrizione dei moduli della Workstation

Menu "Analisi di Fourier"
Il Modulo FFT1 calcola la DFT diretta di un segnale, utilizzando un algoritmo di FFT e permette di visualizzare il modulo (lineare o in dB) e la fase della trasformata.
Il Modulo FFT2 calcola la DFT diretta o inversa di un segnale con un algoritmo di FFT.
Il Modulo STFT calcola la trasformata di Fourier a tempo breve del segnale di ingresso, sulla base dei parametri forniti. Questi ultimi sono, nell'ordine, il tipo e la lunghezza della finestra di analisi, il numero di canali della rappresentazione tempo-frequenza e lo shift temporale della finestra. Il tipo di finestra puo' essere scelto tra quelli comunemente in uso o essere fornito dall'utente, che puo', con l'ausilio del Menu "Operazioni", costruire il filtro di analisi che ritiene piu' opportuno. Il risultato della STFT viene memorizzato sotto forma di una matrice di numeri complessi per le successive elaborazioni.
Il Modulo INVILUPPI estrae dalla matrice di coefficienti complessi della STFT il modulo e la fase dei canali della rappresentazione in funzione del tempo. Questo, da una parte, consente di visualizzare (Cfr. Menu "Grafici") in un grafico prospettico (tempo-frequenza-ampiezza) il modulo della STFT e dall'altra permette di realizzare facilmente un algoritmo di Phase Vocoder per ricostruzioni e modificazioni del segnale di ingresso.
Il Modulo SPETTROGRAMMA prepara una matrice tempo-frequenza con le indicazioni per un grafico a livelli di grigio o a colori del modulo (lineare o in dB) e della fase della STFT.

Menu "Sintesi"
Il Modulo SINTESI ADDITIVA 1 implementa la formula di sintesi relativa al modello armonico dei segnali musicali . Gli inviluppi di ampiezza e di fase (o di frequenza) per ogni parziale sono forniti come funzioni temporali estratte dalla matrice di coefficienti della STFT di un segnale originale (Cfr. Modulo INVILUPPI del Menu "Analisi di Fourier"). In tal modo e' possibile operare la ricostruzione del segnale originale stesso o ottenere sue varianti.

Nel Modulo SINTESI ADDITIVA 2 viene realizzato a parte un insieme di parametri contenente le informazioni relative al segnale che si vuole sintetizzare, cioe': frequenza di campionamento, frequenza fondamentale, lunghezza del segnale, numero delle parziali. Inoltre per ogni parziale sono indicate, per punti (tempo-ampiezza), (tempo-frequenza), le funzioni che rappresentano gli inviluppi. I punti forniti sono interpolati linearmente e la formula di sintesi puo' essere calcolata. Notiamo che il secondo dei due casi descritti e' piu' generale in quanto permette di costruire per Sintesi Additiva dei segnali ex novo, mentre nel primo la sintesi e' intesa nel contesto di un algoritmo di analisi-sintesi particolare, come il Phase Vocoder.

Menu "Elaborazione omomorfa"

Il Modulo DECONVOLUZIONE permette di eliminare l'effetto di riverbero eventualmente presente nel segnale di ingresso, nell'ipotesi in cui il riverbero stesso sia dovuto a repliche opportunamente equispaziate di un segnale originale. Il Modulo in esame calcola e memorizza il cepstrum e il cepstrum complesso del segnale di ingresso; quindi ha luogo il processo di eliminazione del riverbero, che dara' i risultati voluti se il segnale di ingresso verifica l'ipotesi menzionata precedentemente. Nel caso contrario occorre agire opportunamente sul cepstrum complesso individuando e rimuovendo i picchi che danno luogo al riverbero. A tale proposito, e anche allo scopo di poter applicare nel modo piu' generale le tecniche dell'elaborazione omomorfa, sono stati realizzati due moduli che calcolano rispettivamente il logaritmo complesso e l'esponenziale complesso di una data sequenza.

Menu "Operazioni"

I Moduli presenti nel Menu "OPERAZIONI" forniscono gli strumenti necessari per le operazioni elementari del DSP.

Il Modulo GENERATORE permette di generare alcuni segnali di uso frequente nella pratica del DSP.

Il Modulo CONVOLUZIONE calcola la convoluzione lineare di due segnali.

Il Modulo FILTRI implementa alcune operazioni fondamentali del filtraggio digitale. In particolare il Modulo BANCO DI FILTRI fornisce le risposte dei filtri costituenti un banco con le caratteristiche richieste; le risposte in frequenza dei filtri del banco possono essere ottenute con larghezza di banda costante oppure con bande non uniformi a Q costante.

Il Modulo INT/DEC realizza la conversione della frequenza di campionamento del segnale di ingresso, secondo un fattore razionale L/M. Per L (M) uguale a 1 si ottiene, naturalmente, la decimazione per M (l'interpolazione per L) del segnale di ingresso.

Menu "Grafici"

Questo Menu fornisce il necessario supporto per la visualizzazione in varie forme dei segnali

risultanti dalle applicazioni.

3. Sviluppi

Modulo di sintesi vocale basato sull'impiego di filtri FIR

Diverse tecniche di sintesi vocale per formanti, basate essenzialmente sul modello sorgente-filtro, sono state suggerite e studiate da diversi autori. Tutte queste tecniche si basano essenzialmente sull'impiego di risonatori IIR del secondo ordine Questo approccio presenta il vantaggio di un'elevata efficienza computazionale nonche' di una relativa semplicita' nel calcolo dei filtri. L'utilizzo di strutture FIR nei sintetizzatori per formanti e' sempre stato limitato dalla loro bassa efficienza. D'altra parte, le strutture IIR, oltre ai problemi di fase e di ampiezza che intervengono a seconda della configurazione scelta, mostrano una tendenza alla instabilita' che aumenta al diminuire della larghezza di banda. Di conseguenza si ha un notevole aumento della lunghezza della risposta impulsiva dei filtri, che causa a sua volta dei problemi nel raccordo di fase tra due impulsi di pitch successivi. Tali effetti sono spesso estremamente sgradevoli all'ascolto, per quanto in linea di massima i suoni prodotti rimangano intellegibili. Molti dei problemi appena descritti vengono risolti qualora si usino filtri FIR a fase lineare come blocchi elementari. Questa scelta permette di operare con strutture stabili e facilmente controllabili, purche' si superino le limitazioni derivanti dall'impiego di tali filtri. Infatti a causa dell' elevato numero di coefficienti, le applicazioni risultano estremamente dispendiose in termini computazionali e di occupazione di memoria. E' in fase di realizzazione un Modulo di sintesi basato sull'impiego di un banco parallelo di filtri FIR caratterizzato da un'elevata efficienza computazionale. L'aumento dell'efficienza computazionale consegue dalla riduzione dei gradi di liberta' del sistema, partendo da due assunzioni:

1. La produzione della voce puo' essere modellata in modo soddisfacente anche da un sintetizzatore puramente parallelo.

2. Qualunque sia la caratteristica della forma d'onda di eccitazione, il suo effetto puo' essere riprodotto modificando opportunamente i filtri formantici. Di conseguenza la sorgente di eccitazione si puo' ridurre ad un semplice treno di impulsi quasi-periodico.

Sulla base di queste assunzioni sono stati costruiti dei filtri FIR per la riproduzione delle formanti che vengono eccitati in ingresso da un treno di impulsi alla frequenza del pitch. L'operazione di filtraggio si riduce ad una moltiplicazione e da un'accumulazione eseguite solo in corrispondenza di ogni impulso di pitch, in quanto altrove l'eccitazione e' nulla. L'altra notevole difficolta' da affrontare in un simile progetto e' il controllo dinamico dei coefficienti dei filtri. Nel realizzare l'evoluzione di una formante, i coefficienti del filtro relativo devono variare di conseguenza e cio' implica un notevole flusso di dati per descrivere le variazioni

di frequenza e di larghezza di banda. Tuttavia, eliminando un grado di liberta', e' possibile semplificare il problema sfruttando il fatto che l'orecchio umano si comporta come un banco di filtri a Q approssimativamente costante, a frequenze al di sopra di 400-700 Hz. In tal modo si puo' legare la larghezza di banda alla frequenza della formante tramite delle semplici operazioni su una funzione di base che descrive la formante di frequenza minima. Piu' specificamente e' possibile mantenere in memoria un'unica funzione prototipo per la risposta all'impulso che corrisponde ad un filtro passabanda centrato alla frequenza minima ottenibile per una formante. Le formanti a frequenza diversa vengono ottenute decimando con una legge opportuna il prototipo anzidetto. In questo modo si possono variare le frequenze centrali delle formanti cambiando lo step con cui vengono letti i coefficienti della funzione prototipo. Naturalmente sono possibili anche valori di frequenza intermedi grazie ad un campionamento del prototipo molto piu' fitto ed ad una parte frazionaria nel controllo della decimazione. Questo sintetizzatore e' attualmente implementato mediante un programma scritto in C ed e' paragonabile, per quanto riguarda i tempi di calcolo, al corrispettivo realizzato con gli IIR. Si sta studiando una realizzazione in linguaggio assembler del processore TMS320C25.

Una modifica che permetterebbe tecniche di sintesi piu' vicine a modelli percettivi e' quella in cui le prime armoniche del segnale vengono specificate direttamente tramite dei generatori sinusoidali. In tali tecniche, direttamente derivate dal concetto di "banda critica", i risuonatori formantici vengono utilizzati solo a partire da una certa frequenza (circa 400 Hz), mentre l'uso di formanti a frequenze piu' basse viene sostituito dall'opportuno inviluppo delle prime due-tre armoniche del segnale.

BIBLIOGRAFIA

L.R. Rabiner - B. Gold. Theory and application of digital signal processing. Prentice-Hall, 1975.

J.N. Holmes. A parallel-formant synthesizer for machine voice output. in F.Fallside (ed.) "Computer speech processing", Prentice Hall, 1985.

J.N. Holmes. Formant synthesizers: cascade or parallel ?", Speech Communication, Vol 2, 1983.

G. De Poli - A. Piccialli. Dynamic control of FIR filters for sound synthesis. Signal Processing IV, EURASIP 1988. Elsevier Publ.

D.H. Klatt. Software for a cascade/parallel formant synthesizer. J. Acoust. Soc. Am. 67(3), Mar. 1980

P. Simone. Metodi di analisi-sintesi e trasformazioni omomorfe per il trattamento numerico del segnale musicale. Tesi di Laurea, 1991

HYPERMUSIC
software di auto-educazione musicale

Prodotto da: Accademia di Santa Cecilia, Rai Televideo, Regione Lazio
Progetto didattico: Giovanni Piazza
Progetto informatico: Riccardo Bianchini, Maurizio Gabrieli, Stefano Petrarca
Realizzazione: Hyperio - Informatica Musicale

HYPERMUSIC e' un programma didattico concepito per chi desideri "auto-educarsi" in musica con l'ausilio di un calcolatore, e e' composto da tre classi "operative" (Educazione dell'orecchio, Teoria, Composizione), una classe "dimostrativa (Acustica) e una sezione "informativa" (Biblioteca). Le classi di Educazione dell'orecchio e Composizione sono divise in sezione Sperimentale e sezione Scolastica. La sezione Biblioteca fornisce una bibliografia ragionata.
HYPERMUSIC si rivolge a un' utenza diversificata - il destinatario principale e' lo studente della scuola media inferiore e superiore - sia carente di qualsiasi competenza musicale, sia gia' in possesso di una competenza musicale di base.
Al primo tipo di utente il programma offre, mediante le sezioni "sperimentali", un approccio al sistema modale e tonale concepito in forma "ludica" assolutamente empirica e interattiva, esente, cioe', da "help" teorici ragionati, e strutturato in modo da contenere nello schema e nel meccanismo del singolo esercizio tutte le indicazioni necessarie a percorrere un itinerario di apprendimento graduale. La classe di Composizione fornisce procedimenti pilotati, in forme modulari e combinatorie, di costruzione musicale ritmica, melodica, armonica e contrappuntistica.
Al secondo tipo di utente il programa ofre, invece, mediante le sezioni "scolastiche", la possibilita' di sviluppare le proprie competenze fino a un grado abbastanza sviluppato di conoscenza ed elaborazione musicale, includendo "help" ragionati in forma di compendi teorici circoscritti. Gli esercizi di composizione portano fino al completamento dell'armonizzazione di melodie corali con basso dato, ed alla elaborazione di contrappunti fioriti elementari a due parti. Un esercizio sistematico dovrebbe consentire all'utente inesperto di passare, attraverso le prime, alle seconde sezioni.

Nella versione Ms Dos HYPERMUSIC e' stato implementato utilizzando i linguaggi Microsoft C, Borland Turbo C, e il generatore di programmi Matrix Layout, mentre la grafica e' stata realizzata con Publisher's Paint Brush e DeLuxe Paint.
Per la versione Apple Mac Intosh sono stati utilizzati Hypercard e Supercard.
In fase di sviluppo la precedenza e' stata data alla velocita' di esecuzione rispetto alle dimensioni dei file eseguibili e grafici. Questo perche' gli algoritmi utilizzati per il controllo dell'attivita' dell'utente sono, per la musica tradizionale, piuttosto complessi, e si e' voluto che il feedback fosse il piu' veloce possibile.
La versione Ms Dos supporta sia apparecchiature MIDI, sia una delle piu' diffuse schede audio multifunzioni.

CAMUS
UN COMPOSITORE DI STRUTTURE MUSICALI VIA MIDI TRAMITE AUTOMI CELLULARI

MAURIZIO GIRI

Questo programma gestisce fino a 128 voci in tempo reale via MIDI, consentendo un controllo sia interattivo, sia differito su tutti i parametri compositivi.
L'idea che sta alla base del programma è l'utilizzo degli Automi Cellulari come generatori di flussi di dati. Senza entrare nel dettaglio dirò che per "insieme di Automi Cellulari" si intende un particolare modello matematico costituito da molti elementi semplici identici tra loro, che, interagendo, sono capaci di comportamenti estremamente complessi. Per farsi una pur vaga idea di cosa si intenda per "comportamento" degli Automi Cellulari, si può immaginare una colonia di batteri, in cui ciascun abitante cambi, ad esempio, colore in funzione del colore dei vicini. C'è una regola scelta arbitrariamente che determina il cambiamento del colore, e questa regola è valida per tutti i batteri. Facciamo un esempio: una regola può stabilire che un batterio diventa nero se il suo vicino di destra è nero, e diventa bianco se è nero il suo vicino di sinistra. E' evidente che se un batterio cambia colore perchè influenzato dai batteri vicini, influenzerà a sua volta questi ultimi, e la colonia apparirà ad un osservatore esterno come attraversata da configurazioni colorate. Questa non è che una delle - virtualmente infinite - regole di comportamento che si possono applicare ad una "colonia" di Automi Cellulari.
E' stato notato che questo modello sembra cogliere l'essenza di molti sistemi complessi esistenti in natura; ma ciò che interessa in questa sede è la possibilità di utilizzare il mutare delle configurazioni per generare strutture musicali.
In pratica il programma legge le configurazioni come sequenze di numeri. Variando le leggi che regolano il comportamento degli AC è possibile generare sequenze di numeri relazionate, e queste relazioni vanno da una complessità (prevedibilità) minima ad una massima. Ciò che conta è che queste sequenze non sono mai casuali, ma presentano interazioni, ricorrenze e progressioni che vengono trasformate, tramite una elaborata routine di decodificazione nella quale è massimo l'intervento del compositore, in eventi musicali. Dopo l'impostazione delle regole di trasformazione iniziali è possibile intervenire in tempo reale per la variazione delle stesse mentre il programma sta generando la musica. E' possibile anche scrivere una "partitura" che definisca, a priori, tutta una serie di cambiamenti. Per fare ciò ci sono dei moduli, detti supervisori, che

possono variare nel tempo i parametri che controllano le voci, le look-up tables, le leggi di generazioni degli automi e così via. Queste variazioni possono essere definite " a mano", step by step, o controllate dagli stessi automi cellulari.
Oltre alle voci si possono gestire tutti gli eventi previsti dal protocollo MIDI, non solamente la produzione di note, quindi, ma anche la variazione dei controllers, dell'aftertouch, eccetera. Ciò significa che nei sintetizzatori MIDI più evoluti è possibile variare la composizione timbrica o spaziale di un suono tramite gli AC.
E' anche possibile trasformare in tempo reale delle sequenze di suoni eseguite su uno strumento MIDI.
Il programma è stato scritto in FORMULA (una estensione del Forth pensata per la composizione algoritmica) e in Assembly 68000 e gira su Atari ST.
Con questo programma sono stati composti i seguenti brani:
IMPRIMIS, per nastro solo
AMOEBIUS, per nastro, clarinetto e contrabbasso
TABLA RASA, per nastro, clarinetto basso e sintetizzatore
ANIHCCAM, musica per spettacolo di danza
Eseguito ad Arles, Rovereto, Parigi, Monaco.

SUONO 2000 - UNA WORK STATION PER L'EDUCAZIONE AL SUONO

L. M. Del Duca - Un. LA SAPIENZA - Roma
M. Marani - LEONARDO S.p.A. - Massa
G. Bertini - I.E.I. - Pisa

Uno dei settori meno sviluppati e meno curati sulla informatica musicale, rispetto ad altri, è quello della didattica.

A tutt'oggi non esiste un sistema, sia dal punto di vista hardware che software, di riferimento valido per questo.

Questa lacuna è quanto mai grave perché non favorisce lo sviluppo della mentalità e del gusto musicale verso i sistemi elettronici in generale.

Per di più anche nella didattica tradizionale il computer può svolgere una efficace azione di supporto.

È per questo che la Leonardo ha lavorato per la realizzazione di un sistema (Suono 2000) che, se da un lato è predisposto per la didattica non esclude certamente anche applicazioni di carattere compositivo.

La Work Station Suono 2000 è basata su un Personal Computer IBM compatibile (MS DOS), nel quale è inserita una scheda LEONARD-C25 specializzata per l'elaborazione numerica del segnale e corredata di un software suddiviso in unità didattiche per l'apprendimento e la sperimentazione sulla fisica del suono e la teoria musicale.

L'interfaccia utente è resa semplice e divertente dall'impiego di disegni in alta risoluzione e colore, ed è quindi particolarmente adatta per una nuova didattica dell'educazione musicale nella scuola media e nei conservatori musicali.

La Work Station è già realizzata ed è in via di completamento il software didattico, mentre la scheda LEONARD-C25 è già in via di commercializzazione negli ambienti industriali essendo predisposta a qualsiasi applicazione nel campo dell'elaborazione digitale dei segnali; fra queste, degna di nota, è una ricerca in corso presso l'Ospedale Pediatrico Apuano, basata sull'analisi di suoni del torace.

Un'altra importante caratteristica della Work Station Suono 2000 è il basso costo complessivo e la trasportabilità, che la rende accessibile ai singoli studenti ed anche a quelle istituzioni, vedi scuole medie, notoriamente poco dotate di fondi.

PROGRAMMA DIDATTICO DI SUONO 2000

Il programma è suddiviso in n° 5 parti fondamentali:

1. Che cosa è il suono?

In questa parte si analizzano in generale il fenomeno della vibrazione acustica, della propagazione del suono e la maggior parte degli aspetti legati alla fisica acustica, non ultimi gli apparati uditivi e fonatori, ed infine alcuni basilari concetti di psico-acustica.

Il tutto viene affrontato con esemplificazioni grafiche ed ascolto immediato. Ove è possibile si rende interattiva la trattazione come per esempio, la generazione di battimenti, nell'effetto doppler e altri.

Un accento particolare viene posto sulla definizione dei parametri fondamentali quali frequenza, intensità e timbro.

Per quest'ultimo, naturalmente ci si prefigge un approfondimento successivo.

2. Costruiamo suoni con il Computer

Avendo acquisito gli elementi fondamentali della vibrazione sonora, del concetto di periodo e degli altri parametri del suono.

Si è in grado di iniziare a costruirsi i primi suoni partendo dal disegno di forme d'onda arbitrarie e del contemporaneo ascolto di queste.

Creazioni di un archivio di forma d'onda e possibilità di manipolare queste, intervenendo sui parametri fondamentali.

Introduzione del concetto di inviluppo temporale e creazione di eventi sonori complessi, isolati ed in sequenza.

3. Indaghiamo sui suoni naturali.

In questo capitolo la Work Station funge sostanzialmente da campionatore ed è possibile, con un potente editor delle campionature, analizzare le forme d'onde di suoni naturali, confrontarle con quelle di suoni costruiti e manipolarle alla stessa maniera di quelle costruite graficamente.

4. Dai Suoni alla Musica.

Creati i materiali sonori la musica scaturisce dall'organizzazione secondo la propria creatività di questi.

Ed ecco la possibilità di organizzare una partitura via via sempre più complessa.

5. Tecniche avanzate di Sintesi e Analisi del Suono.

Quest'ultimo capitolo è naturalmente una doverosa appendice dedicata ai

musicisti che intendono approfondire le tecniche più avanzate e gli algoritmi di sintesi numerica e di analisi dei segnali sonori.

CARATTERISTICHE HW

La scheda Leonard-C25 è basata su DSP TMS 320025 con 64 KW di memoria RAM programma e 64 KW di buffer dati veloce.

È corredata di convertitori AD/DA integrati in un unico chip TLC 32046 (TI) a 14 bit e frequenza di campionamento fino a 25 KHz presettabile da PC. La scheda colloquia con il PC attraverso una interfaccia parallela ed è corredata di un monitor di gestione residente su eprom veloci.

Il software di gestione su PC è in grado di controllare fino a 4 schede contemporaneamente.

Personal Computer MS-DOS compatibile preferibilmente da 286 Kb ed oltre, scheda VGA, colore e mouse.

REMINISCENZE 2
REMINISCENZE 5
REMINISCENZE 4

ROBERTO MUSTO
Via Ferruccio Zambonini, 23 pal. 4/3
00158 Roma
tel. 06/4513575

NOTE

Le tre composizioni sono state eseguite con un prototipo che realizza in tempo reale, tramite una tastiera, l'evento sonoro e la proiezione di luci colorate attraverso uno schermo.
Il video musicale che presento è la ripresa dal vivo di tale performance.
I suoni utilizzati sono unicamente suoni sinusoidali, la cui frequenza, intensità, attack e decay sono variamente preprogrammabili per ogni composizione.
Ad ogni vibrazione sonora corrisponde una vibrazione luminosa il cui "colore" è in relazione analogica con il suono e la cui evoluzione è parallela al suono a cui corrisponde.
La tastiera, studiata appositamente, ha determinato la necessità di una appropriata scrittura e quindi di una corrispondente nuova "tabulatura".
Per ora la tastiera è dotata di 36 tasti, per cui sono utilizzati 36 suoni e 36 luci colorate per ogni composizione.
Cambiando lo schermo si ottengono strutture colorate ogni volta differenti.

Durata ..... 20'

Data di composizione ..... ott./nov. 1990

Luogo e data di
realizzazione del video ..... Roma, 17 Gennaio 1991

Allegato:

Prospetto generale e partitura di "Reminiscenze 5"

DIECI VIDEOCLIP DI MUSICA CONTEMPORANEA

LORENZO TAIUTI

Il progetto nasce dall'esigenza, da più parti espressa nell'area della musica contemporanea, di riacquistare spazi di diffusione e di spettacolarità. In questo senso si pone primario il rapporto con la televisione. Ed è in campo televisivo che si manifestano alcuni segnali interessanti: da una parte l'enorme sviluppo dell'home-video che investe (e sempre più investirà) la musica videoregistrata, finora presente in rare apparizioni sulle reti nazionali. D'altra parte la videoarte utilizza la musica (contemporanea o popolare) in maniera preminente, quasi a significare una futura internazionalità di linguaggio del rapporto video e musica.
Rapporto già vincente sul piano della musica popolare, che ha conquistato attraverso la formula del videoclip vastissimi spazi di ascolto televisivo. Tutti questi dati emergenti sono i presupposti su cui si muove il progetto "Video Musica Contemporanea".

Modalità del progetto:

1)- Il progetto di "Video Musica Contemporanea" comprende una serie iniziale di dieci Videoclips di 3 minuti ciascuno. Il tempo è pensato per gli standard televisivi, sia quelli del videoclip musicale pop, sia quelli di piccoli programmi inseribili in programmi-contenitore televisivi o come autonomi spazi fra un programma e l'altro, o ancora come compilazione da vendere sul mercato delle videocassette musicali o del videodisco.

2)- Il progetto allinea una serie di autori non per logica di gruppo ma come ventaglio aperto sulla molteplicità di proposte valide nel campo della musica contemporanea, dalla musica elettronica alla musica concreta e ad altre senza limitazioni.

3)- I Videoclips, basati su musiche ed immagini realizzate ad hoc, seguono una linea tematica che privilegia l'oggi, il mondo dei media e dello spettacolo, l'immaginario di massa, i linguaggi creativi della televisione e della Videoarte.

4)- Obbiettivi del progetto sono: la diffusione della musica contemporanea attraverso il mezzo video, sia attraverso la programmazione televisiva sia nella produzione e

distribuzione di videocassette (o videodischi) da parte delle case discografiche nel circuito musicale.
Partecipano alla prima serie di dieci Videoclip di Musica Contemporanea i seguenti compositori:
- Claudio Ambrosini - Giorgio Battistelli - Luigi Ceccarelli - Giuseppe Chiari - Ludovico Einaudi - Lorenzo Ferrero - Daniele Lombardi - Michelangelo Lupone - Salvatore Sciarrino - Luca Spagnoletti

1)- I dieci videoclips sono prodotti presso gli StudiVideo "Eta Beta" di Roma -
2) Le lavorazioni dei video sono svolte su attrezzature di postproduzione "Harry Paintbox" digitale e possono poi essere inviate alla rassegna su ogni supporto richiesto, dal DI al VHS a scelta.
3) I due Video che vi inviamo per la proposta alla rassegna sono "Delitto perfetto" con musica di Luigi Ceccarelli realizzata su computer Atari e campionatore Akai SI000 - e immagini di Lorenzo Taiuti realizzate su "Harry Paintbox digitale" -

Il video "Multietnica e Diffusa" è realizzata da Luca Spagnoletti su computer Atari e campionatore Akai 900 - le immagini sono realizzate da Lorenzo Taiuti su "Harry Paintbox digitale" -
Dei restanti otto video è terminato "Swapping" di Michelangelo Lupone realizzato su computer e per le immagini da Lorenzo Taiuti su "Harry Paintbox digitale".
Sono in via di realizzazione per lo stesso autore per le immagini Lorenzo Taiuti e per diverse tecniche musicali e i seguenti compositori:
- Claudio Ambrosini " Audioclip" (computer e archi)
- Giorgio Battistelli "Ecstasy" (trebatterie computer archi)
- Daniele Lombardi "Dècollages" (tre piani)
- Lorenzo Ferrero "Computer" (piano e computer)
- Ludovico Einaudi "Replica" (computer

I titoli di alcuni dei video possono subire dei cambiamenti.
La totalità dei video sarà terminata per l'inizio dell'estate.

Parte 4: Concerti

I concerti organizzati quest'anno in occasione del IX CIM propongono opere, spesso in prima esecuzione assoluta, di sicuro interesse nel panorama internazionale della musica composta e/o eseguita mediante sistemi informatici.

Le manifestazioni si articolano in due concerti, il cui programma è stato selezionato dal *Comitato Musicale* del Colloquio, preceduti da un concerto di benvenuto a cura del *Comitato Organizzatore.*

Il concerto di apertura, iniziando dall'acustica pura del quartetto di chitarre e concludendosi con una performance con animazione e suono sintetici, vuole unire in un percorso ideale le diverse tendenze compositive della musica contemporanea.

In tutti i concerti, comunque, prevale l'uso di affidare l'esecuzione *dal vivo* di una parte a strumenti tradizionali; a questo proposito, un ringraziamento particolare va a tutti gli Interpreti, solisti di alto livello e di esperienza internazionale, specialisti nel repertorio contemporaneo, che hanno reso possibili questi concerti.

Anche se, per ragioni di spazio, non vengono pubblicate note biografiche a loro relative, hanno certamente l'unanime gratitudine di tutti i Compositori per la competenza e la passione con cui affrontano il loro non facile compito.

Nei pochi casi in cui non è stata tecnicamente possibile la rappresentazione *dal vivo*, viene riprodotta una registrazione su nastro della parte strumentale.

Siamo molto onorati, infine, che il Teatro Carlo Felice di Genova, appena ricostruito dopo quasi cinquant'anni dalla sua distruzione nel corso della seconda Guerra mondiale, e consegnato alla città solo pochi giorni prima della nostra manifestazione, possa ospitare i tre concerti nel suo Auditorium.

Corrado Canepa

Teatro Carlo Felice
Auditorium
ore 21

Concerto di Apertura

mercoledì 13 novembre 1991

Federico Ermirio — *Souvenir d'hiver*
per quattro chitarre

Giuliano Palmieri — *Dialogues*
per fagotto e real-time digital processing e sintesi
prima esecuzione

Corrado Canepa — *Anceps Imago*
per due clavicembali e suoni di sintesi
prima esecuzione

Mauro Graziani — *Mappe di Luce e Suono #4*
per computer

Quartetto chitarristico di Asti:
Marco Silletti, Gianni Nuti, Maria Grazia Reggio, G.Paolo Bovio, chitarre

Bruna Pannella, Elisa Soldatini, clavicembali

Rino Vernizzi, fagotto

Federico Ermirio

Souvenir d' hiver

per quattro chitarre
(1985)

Eseguito la prima volta nel 1985 al Festival Nuovi Spazi Sonori, a Roma, è oggi qui presentato dal Quartetto di Asti, un ensemble di recente formazione, particolarmente interessato al repertorio contemporaneo (ha già inciso per la RAI e per un CD di prossima pubblicazione). Riportiamo una recente nota del chitarrista-compositore A. Gilardino sul brano in programma:

"Ermirio esplora lo spazio sonoro e la gamma timbrica del quartetto di chitarre con una scrittura volta alla ricerca di sottili, allusive sonorità. Il titolo della composizione ne rivela sia il proposito evocativo che il particolare indirizzo espressivo, nel quale si definiscono *climi sonori* nitidi e trasparenti, come glaciati, opposti a momenti di energia contundente e a prolungati flussi di materia sonora: in queste diverse "evocazioni", il compositore fa un uso assai specifico delle risorse strumentali, impiegando i suoni armonici per suggerire il ricordo di ciò che è freddo e lucente, gli spessori accordali per la loro corposa durezza e un dinamicissimo, elastico *rasgueado* per l'algido scorrere di eventi materici."

FEDERICO ERMIRIO (Genova, 1950), compositore, direttore d'orchestra, ha studiato sotto la guida di S. Lauricella, G. Petrassi, D. Paris e O. Suitner, presso il Conservatorio di Genova, l'Accademia di S. Cecilia in Roma e la Hochschule fur Musik di Vienna.

Attivo soprattutto come compositore, ha ottenuto primi premi e prestigiosi riconoscimenti in numerosi Concorsi e rassegne internazionali: Olympia Prize, Guido d'Arezzo, Marinuzzi, Stockhausen, Angelicum, Trio-Basso Wettbewerb, Kucyna Prize, e altri.

La sua musica viene eseguita nei principali Festival di musica contemporanea (Roma, Milano, Torino, Alicante, Amsterdam, Atene, Rotterdam, Boston, Zagabria, Marsiglia, Ginevra, Nizza) e radiotrasmessa da RAI, ERT1, RTSR, e Radio yugoslava e ungherese.

Suoi lavori sono pubblicati da Edipan, Sonzogno, BMG Ariola, Pro Studium, Berben.

Ermirio è attualmente direttore del Conservatorio "A. Vivaldi" di Alessandria.

Giuliano Palmieri

Dialogues

per fagotto e real-time digital processing e sintesi
(1991, dur. 15')

Dialogues è stato scritto per questa manifestazione e nasce da una collaborazione con il fagottista Rino Vernizzi, primo fagotto dell'Orchestra di S.Cecilia.

Il brano è un dialogo tra uno strumento acustico, talvolta "rivisitato" da elaborazioni su computer in tempo reale, ed un'orchestra sintetica gestita dal compositore in tempo reale mediante quattro calcolatori.

Una delle caratteristiche di *Dialogues* è la componente stocastica che emerge in alcuni momenti sia nella partitura per fagotto che in quella informatica.

GIULIANO PALMIERI ha compiuto i suoi studi musicali accanto a quelli umanistici ed universitari (facoltà di biologia), presso i conservatori di Genova e Venezia, dove ha conseguito il diploma di musica elettronica sotto la guida di Alvise Vidolin.
Le sue esperienze spaziano dalla musica intesa autonomamente alla musica per teatro (Biennale di Venezia 1984) e per il balletto (Festival del Balletto di Avignone, 1985). Precedentemente era stato invitato al Fringe Festival di Edinburgo con un lavoro per pianoforte e nastro.
Attualmente collabora con *IRIS* e con il Laboratorio di Informatica Musicale del Dipartimento di Informatica, Sistemistica e Telematica dell Università di Genova;
Tra le partecipazioni più significative citiamo:
- 1987, *Traslot*, Festival Balletto di Nizza;
- 1988, *The Fantastic Universal Frame*, Grenoble; installazione multimediale con l'artista Sergio Pavone;
- 1989, *Fiori Paralleli*, New York, Galleria Maison Des Artistes, installazione multimediale con il pittore Luca Babini;
- 1989, Partecipa con una composizione per elaboratore in tempo reale e performance gestuale al concerto tenuto in occasione del *European Workshop on Artificial Intelligence and Music*, Genova.
- 1989, *V.S.O.P.*, Grazermesse International, Graz;
- 1991, *Full Stop*, Galleria Orti Sauli, Genova; installazione multimediale con l'artista Sergio Pavone.

Corrado Canepa

Anceps Imago

Concerto per due clavicembali e suoni di sintesi
(Andante - Adagio - Allegro)

(1989/90, dur. 10')

Il titolo, in lingua latina, si può tradurre "*figura a due facce*", o "*immagine bivalente*"; l'idea di doppio suggerita dal termine "anceps" può ben descrivere lo spirito di questo brano, tutto giocato sul numero DUE: due sono i clavicembali, e tutta la partitura è continuamente in bilico tra due poli: antico e moderno, tonale e atonale, ritmico e aritmico.
Inoltre, il rapporto duale tra solisti e orchestra è qui ancor più evidente, essendo l'orchestra tutta *elettronica* (da notare che non si è fatto uso di campionamento di suoni "concreti", ma solo di sintesi); così altre dualità possono essere individuate, ad esempio, nella *naturalità* di alcuni suoni di sintesi rispetto alla *artificialità* di altri, ecc.
Le tecniche di sintesi della parte elettronica sono miste, analogiche e digitali; doppio è anche l'impiego del computer: dal punto di vista sonoro, esso gestisce alcune voci, tramite MIDI, ma ha costituito un ausilio alla composizione anche nel momento della stesura della partitura. A questo proposito, il brano è stato descritto utilizzando il formalismo implementato nel sistema software *HARP*, come spiegato nel relativo articolo del presente volume.
La dedica del brano a Valentina, una bimba nata "mentre questo lavoro prendeva forma", ed a tutti i suoi coetanei, tocca un ulteriore aspetto *anceps*: la distanza generazionale tra compositore e dedicatari dell'opera rende inevitabile un loro ben diverso "sentire" il medesimo periodo storico che stanno vivendo e nel quale l'idea del brano è maturata, cioè la fine del XX Secolo; l'opera stessa, nonostante la sua tensione unificatrice tra antico e moderno, finirà per avere per i suoi dedicatari, come tutto "il suono del nostro oggi", "il sapore di un - lontano - ricordo d'infanzia"...

CORRADO CANEPA, nato a Genova, approda ad una sintesi tra musica ed elettronica solo dopo aver coltivato per molti anni separatamente queste due discipline; l'uso dell'elettronica rappresenta per lui uno strumento di lavoro e di ricerca, che risponde pienamente alla sua esigenza di libertà compositiva e gli consente una gestione integrale del processo produttivo, dalla composizione alla realizzazione discografica.
Non tutta la sua produzione musicale è però specificamente *elettronica*, e, comunque, la sua attività abbraccia tutto il campo della Comunicazione Sonora, sia nella didattica (stages, conferenze, progetti con la Scuola), sia nella collaborazione con teatro, danza, video-tape, arti visive, radio-TV.
Presso il DIST - Università di Genova, con R. Zaccaria e A. Camurri ha dato inizio, nel 1984, all'attività del Laboratorio di Informatica Musicale, dove si occupa di descrizione ed elaborazione del suono mediante software, con particolare attenzione ai problemi di composizione assistita dal calcolatore.
Quattro sue opere elettro-acustiche hanno ottenuto riconoscimenti in Concorsi Internazionali.

Mauro Graziani

Mappe di Luce e Suono #4

per computer

"I miei lavori sono costituiti da una parte sonora e una visuale che possono essere considerate come proiezioni della stessa organizzazione formale. Entrambe le parti sono generate dallo stesso personal computer equipaggiato con software di mia creazione, che utilizza gli stessi procedimenti matematici per controllare sia il suono che l'immagine, creando tra loro stretti legami di posizione e movimento. Basandomi su questi presupposti, ho realizzato sia delle installazioni audio-visuali autonome che delle composizioni che prevedono la presenza di un esecutore.
Nelle installazioni il computer è programmato per produrre, in tempo reale, immagini in mutazione continua e suono, lavorando senza interventi esterni come una macchina autonoma in grado di creare di continuo nuova informazione e modificare, istante per istante, l'uscita video e quella audio.
Nelle composizioni, invece, l'elaboratore viene pilotato dal vivo da un esecutore che agisce come controllore dei processi programmati dalla macchina, determinandone lo svolgimento e influenzandone lo sviluppo.
Mappe di Luce e Suono #4 fa parte di quest'ultima categoria. In questo brano, il lavoro di composizione consiste nella realizzazione del software che controlla la parte musicale, quella grafica e determina la loro connessione, nonchè le modalità di interazione con l'esecutore, creando uno o più ambienti esecutivi, ognuno dei quali è caratterizzato da uno spettro di possibilità. Il compito di quest'ultimo è, invece, quello di esplorare tali ambienti esecutivi e costruirsi una mappa di azione per muoversi al loro interno. La scelta dell'esecutore ha quindi una precisa valenza strutturale."

MAURO GRAZIANI (Verona, 1954 - studi musicali al Conservatorio "B. Marcello" di Venezia con A. Vidolin), lavora dal 1970 nell'area della musica elettronica.
Dal 1976 opera presso il CSC di Padova come compositore e ricercatore. Ha compiuto varie ricerche finalizzate all'impiego della tecnologia informatica in campo musicale e composto opere sia di Computer Music che di tipo multimediale, eseguite e radio-diffuse in Italia e all'estero (Europa, America, Est europeo).
Ha ricevuto commissioni da parte del LIMB - Biennale di Venezia (1980, "The Silent God") e dalla RAI (1982, "Trasparenza"). Ha partecipato a manifestazioni della Biennale di Venezia (1980, 1982, 1986, 1989) ed è stato selezionato per partecipare a "Venezia Opera Prima".
Suoi brani hanno ottenuto riconoscimenti al Concorso Nazionale "Città di Abbadia S. Salvatore" (1.o premio, 1985) e al IX e XI International Electroacustic Music Award di Bourges.
Ha collaborato, in qualità di esecutore tecnico all'elaboratore, con A. Clementi ("Fantasia su roBErto fABriCiAni", "Parafrasi"), J. Chadabe ("Canzona Veneziana"), F. Donatoni ("Elettronico") e con A. Vidolin e S. Sapir per la realizzazione e messa in scena del "Prometeo" di L. Nono. Ha pubblicato articoli su varie riviste del settore e tenuto conferenze in varie sedi.
Suoi brani sono incisi sui dischi Edipan INSOUND 2 (PRC S20-16) e INSOUND 3 (PRC S20-18)

Teatro Carlo Felice
Auditorium
ore 21

Concerto IX Colloquio di Informatica Musicale

giovedì 14 novembre 1991

Luigi Ceccarelli	*Preludio al terrore* per nastro magnetico con voce recitante
Roberto Doati	*70 viti da legno* per nastro solo prima esecuzione
Agostino Di Scipio	*Fractus* per viola e nastro
Amnon Wolman	*Etude - hommage a`Bartok #2* per nastro solo
Simonetta Sargenti	*Naturalmente* per viola e sintetizzatore
Giovanni Cospito	*Monocromia VI* per nastro solo
Andrea Molino	*Life Songs* per percussioni e computer (registrazione su nastro)

Hans van Dijck, viola (in *Fractus*)

Pietro Mianiti, viola (in *Naturalmente*)

Luigi Ceccarelli

Preludio al Terrore

per nastro con voce recitante
(1990, dur. 10')

Realizzato nel 1990 su commissione del XX Festival de Musique Experimentale di Bourges, in occasione del bicentenario della Rivoluzione Francese, è concepito come un preludio all'ascesa al potere di Robespierre, culminata poi nel cosiddetto periodo del Terrore. Nel dibattito tenutosi all'*Assemblee*, il 30 maggio 1791, sulla necessità di mantenere o meno la pena di morte, Robespierre tenne un accorato discorso a favore della sua abolizione, sostenendo principalmente due proposizioni: "*La premiere, que la peine de mort est essentiellement injuste; la deuxieme, qu'elle n'est pas plus repressive de toute les peines, et qu'elle contribue beaucoup à multiplier les crimes qu'à les prevenir*". Contrariamente a queste argomentazioni, l'Assemblea si pronuncerà a favore della pena capitale, introducendo l'uso della ghigliottina, della quale, in seguito, contraddicendo le sue precedenti affermazioni, anche Robespierre farà largo uso. Ciò accrebbe la sua fama di fautore delle esecuzioni politiche: in dieci mesi di Terrore si calcola siano cadute 16.594 teste. In considerazione di questi fatti, la rilettura del discorso di Robespierre, soprattutto in alcuni passaggi, si rivela oggi in tutta la sua luce particolarmente sinistra ed inquietante. La voce recitante è di Giovanni Riccioli.

Sistemi utilizzati:
Campionamento: AKAI S1000
Elaborazione: ATARI
Processore di segnale: Lexicon e Yamaha SPX1000
Realizzazione: Studio privato dell'Autore

LUIGI CECCARELLI considera il proprio lavoro più che un'organizzazione di suoni, un processo di organizzazione di eventi complessi nel tempo; in tutte le sue opere il suono viene sempre pensato in rapporto con lo spazio e la dimensione visiva. Perciò egli considera particolarmente importante il lavoro per la danza ed il teatro: dal 1978 collabora con Altro Teatro, ed è anche fondatore di Electra Vox Ensemble, formazione operante nel campo della musica elettro-acustica e del teatro musicale. Benchè sia considerato un musicista elettronico, il suo interesse si rivolge anche alla musica strumentale sia della cultura occidentale che extra-europea. Nelle sue opere impiega di solito tecniche miste, con strumenti elettronici che elaborano il suono strumentale dal vivo. E' membro del direttivo dell'Associazione *Musica Verticale*, e dal 1979 insegna Musica Elettronica al Conservatorio di Perugia.

Roberto Doati

70 viti da legno

per nastro solo
(1990/91 - dur. 4'33")

"Attraverso un percorso casuale (il gioco) ricevo come vincita da un amico cageano 70 viti da legno, che rappresentano la sua visione del compositore di musica sintetica. Come le viti inserite tra le corde del pianoforte preparato di Cage, in quest'opera settanta difoni della lingua italiana sono stati gridati sulle corde di un pianoforte, dando vita a risonanze che, grazie alla modalità di produzione, presentano un carattere vocale.

Questo materiale, naturale o trasformato da filtri, trasposizioni e missaggi complessi, è affiancato da suoni sintetici, scelti in uno spazio timbrico i cui estremi sono: rumore colorato e spettri quasi-armonici. La forma ricalca 4' 33" di John Cage. Essa è infatti costituita da tre parti della stessa durata di quelle del famoso pezzo di "silenzio". La prima (30") è un unico grande crescendo di materiali omogenei. La seconda (2'23") è un movimento centripeto: l'articolazione degli ultimi suoni corrisponde a quella formale dell'intera parte (da una grande densità di materiali non omogenei a una rarefazione desolata). Nella terza parte (1'40") suoni concreti (risonanze di pianoforte, difoni) e suoni sintetici si fondono e si trasformano, fino a divenire creature timbricamente ambigue.

All'interno di ogni sezione, il materiale è stato ordinato utilizzando programmi appositamente realizzati di generazione pseudo-casuale degli eventi sonori.

Ciascun programma è progettato per produrre organizzazioni formali specifiche: sequenze o cluster di suoni molto brevi, fasce che possono essere anticipate e/o seguite, come una sorta di pre-eco o di eco, da suoni che di esse fanno parte, strutture di suoni collegati fra loro da glissandi, sequenze molto rarefatte di suoni lunghi. Nella generazione delle diverse strutture, i programmi hanno un certo margine di scelta, pincipalmente in ordine a densità e durata media dei singoli suoni."

L'opera è stata realizzata su Personal Computer dotato di convertitori AD e DA, con il programma MUSIC 5 nelle versioni del CSC - Università di Padova (Alessandro Colavizza) e L.M.A. - CNRS di Marsiglia (Daniel Arfib).

ROBERTO DOATI ha studiato Musica Elettronica e Informatica con A. Mayr, P. Grossi e A. Vidolin. Dal 1979 lavora come compositore e ricercatore nel campo della psicologia della percezione musicale presso il CSC (Padova).
Nella veste di organizzatore musicale ha collaborato con la Biennale di Venezia (83/86) e dall'88 con i concerti di Repubblica-Ricordi *Eco e Narciso*.
Svolge attività didattica attraverso corsi e conferenze itineranti, durante i Corsi Estivi del CSC e al Conservatorio di Cagliari.

Agostino Di Scipio

Fractus

per viola e suoni generati da computer su nastro
(1990)

Il titolo fa riferimento sia al carattere frammentario e discontinuo della composizione, sia ad alcune procedure usate nella sua realizzazione.

In particolare sono stati usati dei sistemi di equazioni semplici ma non prevedibili, la cui evoluzione alterna fasi di ordine strutturato a fasi di caos, o di ordine nascosto in forme di dimensione non unitaria (frazionaria).

In tale modo si è perseguita una forte (e volutamente irrisolta) dialettica tra materiali di riferimento fisso e sfumate deformazioni di essi, principalmente per quanto riguarda i parametri timbro e densità.

I suoni di sintesi sono stati realizzati mediante computer presso il CSC - Università di Padova nell'inverno 1989/90.

Software:
1) sintesi digitale del suono MUSIC 360;
2) compilazione del MUSIC 360 data score: programmi custom dell'Autore
3) editing dei campioni e missaggio digitale post-sintesi: ICMS

Hardware principale: IBM 3090, IBM PC/AT.

AGOSTINO DI SCIPIO (Napoli, 1962) ha studiato Composizione con G. Bizzi e M. Cardi e Composizione musicale elettronica con M. Lupone presso il Conservatorio de L'Aquila. Ha inoltre seguito corsi di approfondimento con J. Dashow, R. Doati e R. Boesch.
E' autore di concerti e seminari introduttivi alle problematiche della Musica Elettronica e della Computer Music. Collabora con *Musica Verticale* (Roma) e con il *Contempo Ensemble* (Venezia). Svolge attività compositiva presso il CSC (Padova).

Amnon Wolman

Etude - hommage à Bartok #2

per nastro solo
(1984 - dur. 10')

"*Etude* was realized on the System Concepts digital synthesizer at the CCRMA - Center for Computer Research in Music and Acoustics at Stanford University. The synthesizer, known as the *Samson box*, was designed by Peter Samson for CCRMA. It is controlled through software developed by the composer and other researchers and composers at CCRMA, including William Schottstadet's PLA program.
The piece employed among other John Chowning's FM algorithm, and my own FM Flute.
Etude was in 1984 a finalist at the *Luigi Russolo International Competition* and premiere at *Stanford Computer Music Festival.*"

AMNON WOLMAN, a native of Israel, is currently on the faculty at Northwestern University, (Evanston, Illinois - USA), teaching composition and directing the Computer Music Center. He has written more then fifty pieces which include symphonic works, vocal and chamber pieces for different types of ensembles, film music, music for theater and dance, and works involving computer generated or processed sounds.
His recently completed piano concerto was premiered at the *Banff Festival* in Canada by Tony De Mare, and is scheduled for an American premiere by the Minnesota Chamber Symphony in April 1991. He is currently working on a piece for tenor, string quartet and tape, commissioned by Paul Sperry and the Alexander String Quartet.

His list of awards include first prize at the *NewComp International Competition* 1988, third prize at the *Okanagan International Competition* 1982 (grants from the Dutch Government, America-Israel Culture Foundation, the Rockefeller Foundation, the Djerassi Foundation, Meet the Composer NY, Illinois Artts Council and the MacDowell and Yaddo Foundations). He has been commissioned by H. Holliger, H. Sparnaay, and B. Sluchin among others. His music has been played extensively around the world, most recently in *Montreal New Music America'90*, Darmstadt and Eseen (Germany), Madrid (Spain), Buenos Ayres (Argentina), Radio France and other.
His music is recorded by Wergo, iMedia Montreal, and the Stanford University Press, and published by the Israel Music Institute and the Israel Music Publishers.

Simonetta Sargenti

Naturalmente

per viola e sintetizzatore
(1989 - dur. 8'30")

La composizione prevede una base di episodi timbrici che trascorrono l'uno nell'altro e sono caratterizzati ciascuno da un elemento predominante che la viola con i suoi interventi ora intensifica, ora contrasta.

Il carattere della parte elettronica è essenzialmente statico, quindi i suoni, prodotti con sintetizzatore digitale via MIDI, non sono stati molto elaborati, ma piuttosto sovrapposti per creare dei pannelli nei quali è importante soprattutto il carattere dell'evento, ad esempio fascia sonora o effetto particolare.
L'andamento ritmico è dato prevalentemente dalla parte del solista.

Il pezzo è stato eseguito per la prima volta a Munster il 18-1-1989 nell'esecuzione di Ulrich von Wrochem alla viola e dell'Autrice per la parte elettronica; questa sera sarà invece presentato da Pietro Mianiti, membro del Nuovo Quartetto di Roma, docente al Conservatorio di Alessandria e solista di esperienza internazionale.

SIMONETTA SARGENTI ha compiuto parallelamente studi umanistici e musicali a Milano, laureandosi in Filosofia e diplomandosi in Composizione. Si è perfezionata in Musicologia presso l'Università di Bologna; particolarmente interessata allo studio dei problemi relativi alla musica contemporanea, tra i quali, negli ultimi anni, l'applicazione dell'informatica all'analisi musicale, ha pubblicato saggi su riviste specializzate e collaborato a conferenze e dibattiti sulla Nuova Musica.

Dopo aver partecipato a seminari relativi alle tecniche di analisi e sintesi del suono (EMIT di Milano, CSC di Padova) ha composto brani da concerto e musiche per ambienti.

E' docente di Storia della Musica presso la Civica Scuola di Musica di Milano.

Giovanni Cospito

Monocromia VI

per nastro solo
(dur. 4')

"L'esigenza di partenza delle Monocromie è la necessità, da me avvertita, di sedimentare l'esperienza timbrica nella mia produzione di Computer Music.

La percezione del timbro, nell'esperienza quotidiana viene generalmente correlata a sorgenti di tipo fisico ed, in questo, hanno un ruolo fondamentale i giochi di richiami della memoria, che sono soggettivi ma anche storici.

Le Monocromie assolutizzano, nella composizione formale, il punto di vista del timbro: tutti i parametri e la strutturazione stessa degli eventi, sono visti come micro-parti del timbro.

Esse si muovono intorno ad un unico evento timbrico, dilatato con varie tecniche di proliferazione, in cui l'evento primario è sempre riconoscibile ed in cui la memoria ha sufficiente respiro per giocare con i propri richiami e per riportare nella dimensione del vissuto più profondo l'esperienza timbrica elettronica."

Andrea Molino

Life Songs

per percussioni e sistema digitale

Registrazione effettuata all'Auditorium RAI di Torino in occasione della *prima* il 6 10 1991; percussionista Daniele Vineis.

Il brano fa parte di un progetto di più ampio respiro intitolato "Live", complesso di lavori musicali di dimensioni ed impegno progressivamente crescenti, sviluppato in collaborazione con l'Associazione "De Sono" di Torino e sotto la supervisione di Alvise Vidolin per quanto riguarda la parte informatica.

Ciascun lavoro incluso in questo progetto utilizza un sistema digitale misto di volta in volta più sofisticato e complesso: in realtà, è un preciso obiettivo del progetto fare sì che le esperienze e le soluzioni tecniche adottate in uno dei lavori, tanto in campo strettamente informatico quanto più generalmente musicale, vengano poi utilizzate nei lavori seguenti.

Life Songs si compone di quattro sezioni principali, caratterizzate ciascuna dall'uso di strumenti a percussione di materiale omogeneo: metalli nella prima sezione, vetri nella seconda, legno nella terza; nella quarta sezione il suono madre della grancassa fa da supporto al ritorno dei vari strumenti, questa volta combinati insieme.

La prima sezione da sola, con poche varianti, costituisce il brano "Light Metal".

Nell'ambito del progetto "Live" sono già stati realizzati i lavori: *Live Quartet*, per quartetto d'archi e amplificazione; *Places*, per clarinetto solista, 4 clarinetti e live electronics; e *Light Metal.*

E' in corso di realizzazione *Five Live*, per ensemble strumentale e sistema digitale, destinato al Gruppo Musica Insieme di Cremona.

ANDREA MOLINO (Torino, 1964) ha studiato composizione con A. Corghi e direzione d'orchestra G. Taverna a Milano, e musica informatica a Venezia con A. Vidolin.
Lavora al CSC di Padova grazie ad una borsa di studio dell'Associazione "De Sono" di Torino.
Affianca all'attività di compositore quella di direttore d'orchesta, in Italia e all'estero.

Teatro Carlo Felice
Auditorium
ore 21

Concerto IX Colloquio di Informatica Musicale

venerdì 15 novembre 1991

M.Bertoni, E.Serotti	*Monologo1* da New Machine Voice per computer e campionatori (registraz. su nastro)
Cort Lippe	*Music for harp and tape* per arpa e nastro
Francesco Galante	*Lontano* per nastro solo
Ricardo Dál Farra	*EGT* per live electronics (registraz. su nastro)
G.Cospito, V.Simmarano	*Interventi su* *"Intercomunicazione" di B.A.Zimmermann* per violoncello, pianoforte e live electronics

prima esecuzione

Metella Pettazzi, arpa

Guido Boselli, violoncello
Loredana Marcolin, pianoforte

M.Bertoni, E.Serotti

Monologo1

per computer e campionatori in tempo reale
(1989 - da New Machine Voice)

"New Machine Voice" è un progetto di ricerca sulla voce umana che utilizza il sistema digitale campionatore/sequencer in tempo reale. Il progetto sottintende uno studio approfondito sul materiale sonoro utilizzato, e su argomenti extra-musicali come la de-strutturazione morfologica del linguaggio ed un possibile nuovo ruolo per l'attore/performer, *sostituito* e *alleggerito*. dalle macchine.
Gli Autori hanno scelto tre *voci* esemplari nel panorama artistico, campionandole, processandole digitalmente, studiando e catalogando i modi espressivi dei rispettivi interpreti; hanno operato digitalmente centinaia di *tagli* sul materiale vocale, isolando fonemi e intra-fonemi (particelle insignificanti). Successivamente riordinati, i campioni così ottenuti si possono suonare sulla tastiera come su di un *pianoforte preparato*, in nuove ed originali composizioni. Con interfaccia grafica che permette il controllo in tempo reale dei parametri musicali, hanno poi elaborato la partitura MIDI.
Dalla prima fase di questa ricerca sono nati: *Monologo1, Canto* (da/su *Sequenza III* per voce femm. di L.Berio) e *Animale Macchina* (da/su *Cantare la voce* di D. Stratos). *Monologo1* "suona" la voce di Carmelo Bene, tratta dalla registrazione del lavoro teatrale "Manfred". di G. Byron.
Lo sforzo dell'Attore di scomparire dalla scena, annullandosi come essere e ponendosi solo come mezzo per far risaltare la poesia del testo, viene in *MONOLOGO1* applicato alla sua stessa voce con mezzi elettronici, facendo risaltare la poesia della sua voce, libera da significati testuali e linguistici. Il prodotto sonoro dell'uomo diventa *dato*: registrato, catalogato, usabile.
L'uomo non "è" più. La frase, la parola, il fonema, l'intrafonema, i sospiri i suoni sono liberati, rispettando i registri e i modi della "phonè ". La voce diventa autonoma partitura.

Hardware utilizzato:

personal computer	ATARI 1040ST
campionatori	EMULATOR II, EMAX, EMAX SE;
riverberi	LEXICON PCM70, ART DR1;
registratore digitale	SONY EV-S700ES + converter PCM 701ES

MARCO BERTONI (Bologna, 1961; specializz. Jazz con F.D'Andrea) e ENRICO MARIA SEROTTI (Bologna, 1960; laurea DAMS con A. Clementi) lavorano attivamente come ricercatori, produttori e musicisti in uno studio privato di Bologna, collaborando con numerosi artisti italiani. Sono inoltre gestori e direttori artistici del Teatro Villa Aldrovandi Mazzacorati (BO) ed ideatori della "Rassegna di giovane musica *Piccoli Sistemi*".

Cort Lippe

Music for harp and tape

per arpa e nastro
(1990 - dur. 15')

Scritta per l'arpista Masumi Nagasawa, è stata da lei eseguita per la prima volta a Tokyo, Ueno Cultural Center, nel marzo 1990. La parte su nastro è stata creata con il software PATCHER, sviluppato da Miller Puckette, i cui consigli tecnici hanno reso possibile questo lavoro; il mix digitale su nastro è avvenuto all' IRCAM di Parigi, con la valida assistenza di Xavier Bordelais.

Il brano si divide in quattro sezioni principali. La prima è definita da *tremoli* ed estremo impiego di registri. La seconda utilizza un ripetuto *sforzando* nel registro più alto, per disgungere gesti che esplorano seconde discendenti, maggiori e minori. Arpeggi di corde *stoppate* caratterizzano la terza sezione, e l'uso di due strati ben distinti e di sorgenti secondarie di suono costituisce la quarta sezione. Questo è il terzo lavoro scritto da Lippe negli ultimi anni in cui l'arpa ha parte preponderante, perciò egli tenta in questo brano di esplorare il gesto musicale e l'espressione per l'arpa in modo differente.

Il nastro è interamente costituito di suoni di arpa digitalmente trasformati ed elaborati. Tutti i suoni usati sono tratti dalla parte strumentale della composizione. Così il materiale musicale e sonoro è lo stesso per la parte strumentale come per il nastro. Il rapporto strumento/macchina è simbiotico, strumento e nastro sono equivalenti nel dialogo musicale. A volte una parte può prevalere, ma nella struttura formale complessiva è implicito un *duo*. (Si potrebbe immaginare il brano come espansione o amplificazione di un *a solo* - il nastro diventa semplicemente un' estensione dell'interprete; ma l'intenzione dell'Autore è invece di creare proprio un dialogo, un *duo* musicale).

CORT LIPPE (1953, USA) studied composition with Larry Austin. He spent three years in Utrecht, The Netherlands, at the Instituut voor Sonologie and worked with G.M. Koenig in the fields of computer and formalized music. Presently, he lives in Paris where he worked for three years at the CEMAMu - Centre d'Etudes de Mathematique et Automatique Musicales, directed by I. Xenakis, while following Xenakis' course on formalized music at University of Paris.

For the past three years he has been employed at the IRCAM (Paris), where he heads 4X System musical applications; and is active as a composer in the Paris-based group NAME (New American Music in Europe), which presents concerts of contemporary American music in Europe.

He as written for all major ensemble formations; his works have been premiered in North and South America, Europe and Far East, and have been presented in numerous radio and television broadcasts. He is recorded by CBS-Sony, Neuma and CAB Records.

Francesco Galante

Lontano

per nastro solo
(1985)

Realizzato presso i laboratori SIM - Società Informatica Musicale di Roma, *Lontano* è stato sintetizzato mediante sistema digitale progettato e realizzato dalla SIM, basato su DSP TMS32010 Texas Instrument. Il controllo in tempo reale è avvenuto mediante personal computer. La tecnica di sintesi impiegata è additiva - tabelle sinusoidali.

Lontano è il primo di una serie di lavori musicali elettronici, scritti da F. Galante negli ultimi anni, che indaga sulla costruzione di micro-eventi sonori che si muovono nello spazio della Banda Critica.

Il movimento continuo del suono e la sua collocazione nello spazio d'ascolto, sono elementi chiave per la comprensione del pensiere sonoro-musicale del progetto stesso, che più che organizzarsi attraverso oggetti sonori, è esso stesso un oggetto sonoro unitario e complesso, che si evolve all'interno di quel confine percettivo, sottile, in cui pulsazione ritmica e timbricità si incontrano.

FRANCESCO GALANTE (1956) vive e lavora a Roma. Ha studiato composizione musicale elettronica con G. Nottoli a Frosinone e frequentato i Corsi Internaz. di Musica Elettroacustica al GMEB di Bourges, con D. Keane, P. Boeswillwald e G. Baggiani. Ha collaborato col *Centro musica sperimentale* di D. Guaccero e svolto attività organizzativa concertistica come responsabile dell' Associazione *Musica Verticale*.

E' co-fondatore della SIM, dove all'attività di compositore affianca quella di ricerca nell'ambito delle tecnologie avanzate per la progettazione di calcolatori veloci, destinati alla produzione musicale in tempo reale; in particolare ha lavorato a circuiti VLSI per la sintesi del suono.

Nelle sue composizioni approfondisce lo studio di fenomeni percettivi legati alle cosiddette "Bande critiche" e sulla possibile loro organizzazione in sistemi musicali.

Particolarmente interessato alle problematiche del rapporto suono/spazio, ha scritto articoli e saggi sull'argomento e proposto soluzioni tecniche per l'ascolto della musica elettronica.

Sue composizioni sono state eseguite e radiodiffuse in Europa e Americhe. Ha inciso per la Fonit-Cetra.

Ricardo Dal Farra

EGT

per live electronics
(1989, in collaborazione con P. Soderberg)

Registrazione dal vivo (agosto 1989) durante il concerto alla *Musikaliska Akademien* di Stoccolma; l'esecuzione alla chitarra di Peter Soderberg è stata elettronicamente elaborata in tempo reale da Ricardo Dal Farra. EGT ha ricevuto una menzione nella categoria "Live Electroacustic Music" al XVIII Concorso Internazionale di Bourges nel 1990

RICARDO DAL FARRA, nato nel 1957 a Buenos Aires , è professore di musica elettroacustica al Conservatorio Municipal de Musica della città e direttore del *Laboratorio de Musica Electronica* della *Societad Argentina de Educacion Musical.*

Membro dei supervisori di *Interface, Journal of New Music Research* (Netherlands), e corrispondente regionale per il Sud America della Computer Music Association.

Dal 1988 cura la serie radiofonica di musica elettroacustica e per computer alla Radio Municipal di Buenos Aires.

Ha studiato presso il CCRMA - Center for Computer Research in Music and Acoustics - Stanford University, USA, ed è stato ospite del CSC di Padova.

Come live electronic music performer ha suonato in Svezia, USA (Sheridan Opera House di Telluride, Colorado, e Center for Fine Arts, Miami) e in Argentina (Teatro Colon, Buenos Aires, Instituto Cooperacion Iberoamericana, Centro Cultural General San Martin). Sue composizioni sono state eseguite in USA, Brasile, Hong-Kong, Svezia, Francia, Bolivia, Spagna, Canada, Colombia, Germania, Mexico, Australia, Italia.

Giovanni Cospito, Vittorio Simmarano

Interventi su
"Intercomunicazione" di B.A.Zimmermann

per violoncello e pianoforte

Il lavoro musicale consiste nella strumentazione con suoni sintetici di un brano scritto per strumenti acustici. Il termine strumentazione si intende nel suo significato tradizionale: affidare una voce ad uno o più strumenti che la connotino timbricamente.

In considerazione delle problematiche poste in ambito musicale dalla strumentazione per strumenti acustici e considerando le possibilità di manipolazione dello spazio timbrico offerte dal computer, si è realizzata una strumentazione che parte dalla simulazione degli strumenti acustici, estrapola i parametri di controllo considerati musicalmente efficaci e, attraverso la manipolazione di questi, disegna spazi di variabilità timbrica.

Si sono così potute realizzare forme di mutazione timbrica innestate con continuità sui suoni degli strumenti acustici, e varie traiettorie di transizioni timbriche.

Finito di stampare
nel mese di novembre 1991
presso la Prima Coop. Grafica Genovese

Il *Colloquio* vuole rappresentare un punto di incontro, di scambio di esperienze fra le diverse realtà dell'informatica musicale e costituisce inoltre un'occasione per la diffusione delle attività di ricerca e di produzione musicale, scientifica e industriale, sia a livello internazionale che locale.

Alcuni importanti temi scientifici, quali per esempio le reti neurali, l'intelligenza artificiale e le scienze cognitive, trovano ampio spazio in diversi capitoli del presente volume. In particolare, si è deciso di strutturare gli atti del convegno in quattro sezioni distinte. La prima, dedicata agli interventi dei relatori invitati, ha lo scopo di fornire punti di vista autorevoli e di scoprire le linee guida principali in alcuni tra i più rilevanti settori dell'informatica musicale. La seconda parte raccoglie i contributi scientifici: dall'elaborazione numerica di segnali, ai sistemi dedicati, all'intelligenza artificiale, agli algoritmi e ai metodi formali per la musicologia. Questa sezione comprende anche un capitolo sui più significativi risultati ottenuti dalle ricerche sulla "Stazione di Lavoro Musicale Intelligente", del *Progetto Finalizzato Sistemi Informatici* e *Calcolo Parallelo*, del Consiglio Nazionale delle Ricerche, al quale lavorano due unità di ricerca: i laboratori di informatica musicale dell'Università di Milano (LIM-DSI) e del DIST di Genova. La terza sezione è dedicata alle comunicazioni, ai rapporti di attività, e alle descrizioni di sistemi per l'informatica musicale e il multimediale. Infine, la quarta e ultima parte contiene le informazioni essenziali sui brani presentati ai concerti.

GRAFICA: MAURIZIO VENTO