Kalev Tiits

# On Quantitative Aspects of Musical Meaning

## A model of emergent signification in time-ordered data sequence

*Academic dissertation to be publicly discussed,*
*by due permission of the Faculty of Arts at the University of Helsinki*
*In lecture hall P3, Porthania, on the 17th of August, 2002 at 10 o'clock*

Printed by Yliopistopaino Oy, Helsinki, 2002

ISBN 952-91-4913-1 (Print)
ISBN 952-10-0621-8 (PDF)

# Preface

This study introduces some approaches to computer-based and other highly formalized methods for analysing form in time-dependent, music-based data. The data in question are defined as a stream of events (elements, signs, samples) that constitute a time series, and contain such structural cues in time which can be experienced as musically relevant. Various classical computer music methods and data representations are discussed and a new method sketched, which draws ideas from recent scientific developments. Different technologies are evaluated in light of their relation to musical meaning, particularly in its symbolic function. A key aspect of this study is that a data-driven processing method is preferred over the more widely used rule-based ones. This study also surveys isomorphisms and connections of sub-symbolic data processing to the general emergence of musical signification, as seen from a semiotic perspective on the reception of music. The study incorporates computer software written by the author, which was used in testing and experimentation. A series of empirical experiments was conducted on three works selected from the solo flute repertoire of the 20th-century. In sum, the present study brings together several threads of thought, including schools as theoretically distant from each other as philosophical-semiotic explication, on the one side, and computer-based, sub-symbolic signal processing on the other.

# Acknowledgements

A work like the one at hand is invariably a product of many ingredients. Consequently, I have been greatly indebted to many people. Particularly I would like to mention professors Teuvo Kohonen of Finnish Academy of Sciences and Eero Tarasti of University of Helsinki, who have been prominent mentors during this study, often indirectly, teaching the principles of scholarly inquiry by personal example. Besides their part, this work has been significantly influenced by the collegial, friendly and extremely knowledgeable comments of Jose Luis Martinez, which helped me in focusing the research problem and methodology in the beginning.

In the workgroups I have had a chance to be part of, there have been many who have contributed to this work by providing good atmosphere and support. In particular, colleagues Pauli Laine and Kai Lassfolk from University of Helsinki electronic music studio, as well as Andrew Bentley and Philip Donner at Sibelius Academy Centre for Music & Technology encouraged me in different stages of this work. Mauri Kaipainen helped particularly in finding financial aid for the project. The writing process inevitably mingled with my other work. Thus, I would like to thank also fellow musicologists at University of Helsinki and both staff and students at Sibelius Academy CM&T for both fun and the hard times together experienced.

Richard Littlefield went through the text, correcting many unorthodox expressions to proper English and thanks are certainly due to him. Big ta!

My work has received support from Wihuri foundation, Emil Aaltonen foundation, University of Helsinki, Niilo Helander foundation, Oulu Conservatoire and Sibelius Academy, either financially or by providing computing platforms and other resources necessary for carrying out this study. I am grateful to all these parties.

Writing this study has not been an easy process, and I wish to give credit to the person who has in a very real way experienced some of the hardships of the work – my wife Helena. I have yet to conclude, whether this would be the right time for thanks or an apology. All the same, the project is ready now, yet perhaps on some journeys, more important than the destination may be the route that took one there. I would like to think this has been one such journey.

Kalev Tiits
Helsinki, 27th June 2002

# Index

# 1. Can a theory of computer-driven, connectionist music analysis contribute to discussions on signification and meaning in music?

One motivation for this study is to integrate scientific ideas drawn from disciplines that are not usually studied together. There has long been a need for integration among music-oriented disciplines, and this need has grown over the years. This need became apparent during my discussions with music-scholar colleagues and during seminars and conferences. It has been a constant cause of dissatisfaction to watch even experts in the field have difficulty understanding each other, let alone take interest in each other's work. During my long and ongoing love affair with computer-assisted study and modeling of music, the disintegration of scientific inquiry has become particularly obvious to me. Not only is it sometimes difficult to communicate with representatives of traditional schools of music analysis, but one finds a lack of clarity in concepts even among the relatively few scholars specializing in computer-assisted musicology. This situation often leads to unhealthy skepticism and wariness toward research paradigms other than one's own.

The modest objective of this study is to make it easier for us to come out of our paradigmatic fortresses and communicate as academic "brothers and sisters in arms". One should of course not be too ambitious here. It is quite unreasonable to think that a single treatise would put an end to fragmentation even in a small subset of the scientific community; it is not so easy to replace disintegration with unity. But even if the present study is just a small step in that direction, it is yet one worth taking.

In pursuit of integration of thought, I am seeking to bring together concepts from various sides of various fences, and to find a common ground among them. First, attention will be directed to the concept of musical meaning as theorized by Leonard B. Meyer. Secondly, we shall look at ideas concerning signs and how they function in communication, focusing on the work of Charles Sanders Peirce, father-figure of the "American school" of semiotics. A third major ingredient of this study is connectionism, parallel distributed computation, artificial neural networks, or unsupervised learning – whichever label the reader finds most appropriate for this special branch of computer science, which stems from the work of many different people, all of whom cannot be mentioned here. The ideas of Teuvo Kohonen, author of the widely known self-organizing feature map, were chosen as the core of the computing architecture used in the experimental part of this study. The reason for this choice lies in the simplicity and elegance of Kohonen's thought, as well as in a more personal motivation: in working with Kohonen, I have come to appreciate his creative spirit. Peirce's and Kohonen's ideas will be the starting points for the present inquiry.

These starting points are rather hefty ones, to say the least. The massive amount of Peirce's writings is particularly intimidating. It would be rash to attempt to apply to music all his views about the world, science, and signification. Instead, emphasis will go to his categorization of signs, especially as it is applicable to my subject matter.

The emerging applications of Kohonen's self-organizing feature maps, and the literature describing them, seem to grow day by day. For this study, a custom-made piece of software was developed, dedicated to our purposes only, with which it is possible to observe all mechanisms of the model at close range and to study its workings from the inside. Implementation of the self-organizing feature map also enabled the empirical part of our study. Access to a computer program at the source-code level was essential for our purposes. Maximum control and understanding of the results of the test-runs could thus be maintained. Moreover, having complete control of software simulation provides instant access to any new concepts that might emerge from the tests, and enables the observer to distinguish side-effects and by-products from real phenomena.

As concerns musical signification, a key concept in this study is Leonard Meyer's notion of *embodied meaning*. This kind of meaning, typical of nonverbal communication, takes place in our reception of music, mostly without our being aware of it. According to Meyer, embodied meaning is crucial to musical signification – an essential part of the cognitive process of musical thinking.

These three sources of ideas are examined against the background of computer-assisted musicology. The latter began in the 1960s as an interesting trend in the mainstream study of music but soon developed into a discipline of its own, marked by an enthusiasm for the use of numerical methods. Typically, computer-assisted musicology borrowed practices and theories from the growing body of work on computer methods in the humanities, such as linguistics and semiotics, as well as from general computer science, cybernetics, and statistical mathematics, just to mention a few sources. Yet it now seems possible that appreciation of computer-assisted musicology as an independent discipline might vanish in the near future. Today, around the 50th anniversary of the first digital computers, one feels a bit uneasy trying to define a discipline, other than computer science, mainly by its use of computers. After all, not many of us would have lengthy discussions about computer-assisted cooking, though we might turn to the Internet to look up an occasional recipe for a guest's favorite dish. Perhaps (and one hopes) "computer-assisted musicology" will be replaced by another term. Some scholars have already started to talk about "new systematic musicology", which includes numerical methods, particularly in their relation to new developments in cognitive science. Only time will tell whether this will become a generally used concept. We may see the end of computer-assisted musicology as its own branch of study, see mainstream musical studies embrace it again, or perhaps see it transformed and adapted to some other definition, methodology, or name.

Now it is time to introduce the main theme of the present study. Throughout this work, the reader is invited to ask the following questions. Is it just coincidence that Meyer's definitions of meaning and Peirce's views on iconic semiosis seem to hinge on concepts much like those found in sub-symbolic computing theory, and particularly in connectionist-oriented work on technical pattern recognition? Or might there be some deeper relationship between these faculties of thought? If so, might such a relationship serve as foundation for interdisciplinary integration?

# 2. A quantitative way of thinking about music

## 2.1 Dynamic musical objects

### 2.1.1 Computer-based segmentation of musical time-sequences

Automated segmentation of the musical time-continuum has not been of major interest in computer-aided musicology, to judge by the number of research papers on that topic. Nevertheless, it is a challenging and interesting sub-division of the field, and the ramifications of such research can be far-reaching and connect in several ways to other sciences, such as semiotics, psychology, cognition research, and applied mathematics. With the music industry's development of MIDI and sound-processing systems, good solutions to segmentation problems are also beginning to spark commercial interest.

The difference between applied technology and basic research manifests itself clearly in segmentation problems. Whereas commercial applications can successfully rely on a heuristic system set up for only one particular purpose, finding a theoretical basis for a general segmentation methodology of time sequences still remains a challenge.

Various segmentation strategies for musical data in share common problems, at least in terms of the "active" nature of the research object: musical data streams consist of dynamic objects, which are characterized by a "refusal to stand still and be pinned down" (Agawu 1991: 78). The analyst of such data deals with manifold pattern variations. The construction of such variations may not follow a simple, limited set of rules; rather, it probably engages with the freedom of every possible appearance and form in which a single musical idea may manifest itself in a work of art.

This study occasionally refers to the notion of *musical object.* The purpose of this is to break free, at least partially, from traditional concepts used in analysis of musical form, such as motif, theme, phrase, period, and so on. A practical distinction is made here between a musical object and its *instances.* The object itself is the *idea* of a theme or a motif at an abstract level. The object thus belongs to a kind of "Platonic world of ideal forms". The object connects to real events, as represented in a musical score, through an interpretive process that brings into existence its instances – or appearances – in written form or as an acoustic sound-flow in time. The object lives a dynamic life cycle, reappearing in various forms across the continuum of a work or a performance. Its varying appearances manifest the subtle organic processes of the object. (This description of the musical object may seem slightly arbitrary, but I shall attempt to justify it by showing what can be built upon such a description.) In any case, the notion of a motif or other musical "idea" manifesting itself in a variety of guises in a work is a commonplace, both in music studies and elsewhere. The more or less subtle variation of repetitive patterns is a universal feature in the psychology of communications (Monelle 1992: 69-70).

Computer analysis of form, which takes into account the dynamism of musical objects, can be thought of as a two-fold undertaking. The first stage consists in finding the pattern boundaries; this is the *syntagmatic* part of the analysis. The system needs to find out what the patterns are in terms of a linear, beginning-to-end analysis. This is a difficult task, because patterns in real musical situations often escape definition – the vocabulary and syntax of a piece are in many cases unique. At the second stage, the detected patterns need to be compared and classified in a *paradigmatic* way. This comparison is meant to reveal the compositional structuring of the work. Both tasks are carried out at many levels of musical representations, and both are non-trivial.

Syntagmatic recognition and isolation of patterns, in its technical implementation, requires fuzzy or adaptive data processing – something going far beyond familiar Boolean schema of computer processing, wherein a statement is always valued as either logically true or false.  Concerning music, it is likely senseless to talk about “statements” at all, and assigning truth values to musical passages can hardly be justified.  The need for adaptive and flexible processing methods is universal in communications, particularly in musical communication: the dynamic or fuzzy nature of musical objects will express itself both in flow- and event-based musical data.  Thus, segmentational analysis of time sequences in general is a profitable place to test and further develop recent technologies in adaptive computing.

In traditional analysis, segmentation often plays a minor role.  If we consider methods such as functional analysis or Schenkerian, reductionist analysis, and compare them to what can be done with a computer, one notes how easily the human brain performs the task, in contrast to the difficulties which developers of automatic pattern-recognition systems are experiencing. Though different analytical schools coexist, there is in most cases a clearly understandable basis for them, even without a full consensus among scholars. In most cases the discussion turns around preferences over several more or less valid alternatives, and the scholar must be able to consider different, viable alternatives, while abandoning the obviously nonsensical ones.  The numerous and interwoven signifying relationships in music always leave room for contrasting views, which, even if they do not agree, must at least allow room for each other to be able to serve as a basis for discussion.

The case is rather different with computer analysis. In the latter, a task that is trivial for a musically educated person becomes a research problem of its own.  One reason for this is the explicitness of computer systems, such that the grounds of analysis need to be specified quantitatively and step by step, in order to achieve to the level of precision required for algorithmic implementation.  One might consider as one way to make a discipline such as musicology more “scientific”.  Potentially, however, this could lead to gross misunderstandings about the role of computing in the humanities: that quantitative results might be taken as inherently right.  The latter conclusion is incorrect, of course, since quantitative does not equal “true”, and numerical methods still require discussion about grounds and presuppositions of the theories that they are designed to prove or falsify.  Not taking the grounds of analysis may have been one cause of the bias against computer methodologies in the humanities.  In computer-based empiricism it is all too easy to take an ad hoc principle and expand it, without asking necessary questions about the soundness of the initial suppositions. The foundations of algorithmic processes are rarely visible from the surface of computer programs. Even when an integral report of the principal algorithm exists, there may be numerous details of implementation, which may or may not have a pronounced effect on the results of an empirical experiment. Even worse, the operational logic of an algorithm may be embedded inside of a computer program in such a tangled way that it is not easy to see or to validate. In such a case, not only it is hard to test or discuss the value of the approach, but it is also hard to test the system at the practical level, even for elementary programming errors.

Another reason for the non-triviality of segmentation research is that computer-aided analysis has been attempted with a number of different methods, which share only a few common principles of operation. It is not uncommon to pick a method from the tool kit of some other discipline adapt it to forming theories about music. In such cases, special attention must be paid to specific characteristics of musical communication in order to arrive at theories and methods of serious scholarly value.

Segmentation would involve the search for and classification of manifestations of objects, as discussed above, at various levels of musical form. The identification of such manifestations allows one to glimpse the ideas behind the audible instances, or “real” musical objects. The processes by which these instances take shape allow one not only to categorize better the nature of the objects, but also to understand the dynamics of the work. Such understanding enables one to move from questions concerning “what” to those concerning “how”. In this way, the researcher works toward the dynamic processes that make a piece of music particular and identifiable. This analytic procedure would explain both the essential details of musical handicraft and, more generally, the *meaning* of the musical work.

### 2.1.2 On meaning in the context of musical objects

There is abundant literature on meaning in music and culture. It includes a great variety of ideas and disputed issues, including the propriety of using the concept of “meaning”. To highlight briefly some major points on the subject, let us start from an intuitive, “pre-scientific” notion of meaning as the *message* conveyed by a musical work. On this definition, meaning has two major aspects: *transmission* and *significance*. Successful transmission requires some commonality in “alphabet” between the two sides of the communication chain. Significance arises from the dialogue between message and available alphabet, though the message can also be used to expand the alphabet.

Various kinds of messages are transmitted in a listening situation. Some of them can be verbalized and studied with scholarly methods. Others may be “purely musical” ones; i.e., they may be explained in the framework of music theory, but are otherwise hard to approach verbally. Some meanings may be of a narrative nature. Narrative elements are most evident in programmatic music. More abstractly, narrative elements can be found in a large body of western music, as well as among other music cultures of the world. Still other types of meaning serve mythical levels, drawing from the largely unspoken mythical content of a particular culture. Such meanings can be properly understood only in their own semiotic context, and do not fall within the scope of the physical, structuralist, and computation-oriented nature of the present study.

Rather, we shall concern ourselves with *immanent* messages, that is, the ones limited to the inside of a musical work or passage. Hence, many common types of signification will not be considered here. Among these are all references and allusions to extra-musical objects, ideas, and even to other musical works.

A message may be thought of technically as the information content transmitted by the work. Information in musical context has been defined as the “non-confirmation of expectation” (Bent 1987: 58). There must be expectations before information can be transmitted. At minimum, there must exist a human or technological receiver that expects something, no matter how vague the expectation might be. The more specialized and distinct the expectations are, the better they will serve the transmission of information.

Limitation to immanent communication is a strict limitation indeed. The importance of immanent messages may even be questioned – after all, in the history of western music we are studying a tradition more than a thousand years old, rich in intertextual references – rich almost to the point of saturation at times. When a student learns about music, she learns about tradition. Music theory has grown from practice confirmed by tradition. Is it possible to limit oneself to immanent communication and still extract interesting information from a musical work? I would argue, Yes, because even without intertextual, mythical or other cultural references, expectations can still be established within a work of art. To raise expectations, the course of action would then be to use simple formal concepts of communication theory: redundancy, entropy, repetition, similarity, periodicity, pseudo-periodicity, and perhaps other, similar general structuring principles.

There is another value in researching immanent messages. Even though such research may seem crude when viewed in the light of sophisticated cultural communication, it has a great advantage that makes its study worthwhile. This is its generality. Music-theoretical concepts and knowledge about historical developments are style-specific, whereas general information structuring principles are not. Amidst the post-modern Babel of artistic subcultures, tribes, and styles, and the even more numerous mixtures of them, to have tools free of stylistic associations can be of significant scholarly value. Moreover, it brings one to the interesting question about the reliance of a particular work on style and tradition in musical communication – to what extent it is generated within a particular work. Immanent messages are, because of their abstract quality, almost paradoxically limited communicatively, but general in the cross-stylistic and cross-cultural sense.

### 2.1.3 Meaning – a vague concept?

Charles Morris (1971: 121-122) correctly pointed out that the word “meaning” is problematic in its vagueness:

> ... for the major purposes which the everyday languages serve it has not been necessary to denote with precision the various factors in semiosis – the process is merely referred to in a vague way by the term “meaning”.

Apparently Morris would prefer to dispose of the word altogether in scholarly contexts. Some of the more precise replacements, such as "language", "syntax", "truth" and "knowledge" are, however, not available in the case of immanent signification of musical messages.[1] On the other hand, there exists a considerable body of work on meaning in music, from authors such as Leonard B. Meyer (1956, 1967), which allows one to approach the term with a little more precision than would be the case, say, with verbal communication. It makes one wonder whether or not there is a significant difference between language and musical communication.

Meaning in immanent communication has an emergent quality. This is to say, the significance of a message grows from initial obscurity into existence via the message – the artistic communication. The emergence of meaning is a process of *interpreting* the message. This takes place in the cognition of the receiver. Interpretation is made possible by an internal web of relationships in the work. It forms a kind of scaffolding or structural frame indicating the processes of musical objects during their life span.

One might suppose that such a cognitive process is highly personal and hence not available to systematic study. My argument is, however, that even though this process is personal, it still follows general rules. Those rules may be studied, and they may be systematic, even to the point where a computational model can be applied. Were it not so, discussion about music in general would be pointless. Yet we carry out such discussions both in everyday and scholarly levels. Consequently, there must be cognitive laws shared by most of us that have to do with the perception of music. In the following chapters, the objective is to find some general behaviors and functions related to information-processing, which are framed according to the nature of the cognitive system. In order to discuss the various aspects of meaning, it is necessary now to set aside the more personal aspects of its emergence, and consider meaning as the primary significant message interpreted *within* a chosen musical work, independent of extrinsic information.

It seems obvious that language and musical communication have slightly different mechanisms for the emergence of meaning. It is generally doubtful that the internal dynamics of a message should be regarded as the primary source of meaning in the case of linguistic communication. Is it not the culture that forms the environment for the message and provides a framework for any signification? Culture certainly is the basis of most linguistic significance. If music were regarded as a language, this would be the case with music, too.

[1] Though musicians often talk about musical syntax and vocabulary, these terms are not usually meant in as rigorous a sense as they are used in general linguistics.

However, occurrences of the symbolic function in music are few and far between in most musical cultures of the world, and in the music of western society this is particularly true. Thus, attempts to concentrate on symbolic communication do not usually produce a very interesting musical experience – to do so cannot capture our attention for extended periods of listening, even though this mode of listening is sometimes offered to children in the form of "musical fairy tales". When the special character of musical communication is considered, for an adult listener the immanent signification probably dominates (Meyer 1956: 35-38). Meyer refers to signification arising from a situation in which music is pointing at music as *embodied meaning*; symbolic, assigned meaning he calls *designative meaning.* The term "signification" is much used in the context of designative meaning, and less so in that of embodied meaning. In this way, "meaning" can be a suitable term by which to denote a wider range of types of semiosis than does the term "significance", which is often more closely associated with designative meanings.

As used here, the word "meaning" allows for a certain amount of detachment between the semiotic meta-language of language and the meta-language of music. The underlying notion, which is also a defense against Morris's criticism, is that linguistic terms and concepts are not always translatable as such to musical communication. Thus, the liberty is taken here to use the term "meaning" in a slightly wider sense than "signification". A distinction between "meaning" and "signification" will be further elucidated by an examination of the *sources* of meaning or signification, which are present in a situation of musical communication. More discussion of the matter will follow, in the context of application of semiotic theories.

### 2.1.4 Abraham Moles' theory of information and meaning

French scholar Abraham Moles takes a different point of departure for his theory of meaning. He builds his concept of the latter on information theory, and his work *Information Theory and Esthetic Perception* (1966) is considered one of the basic texts on art and information theory. Along with examples of musical signification, he also illustrates signification in language and cryptography. Despite his different background and point of departure, Moles (1966: 62-65) ends up with a concept of meaning not unlike that of Meyer. He draws a strong association between meaning and structure, using the concept of information: "... [the] absence of natural meaning is connected with an overly large flow of information: in the current psycho-esthetic sense, with the absence of structure, that is, of internal organization."

In the Foreword to the English edition of his book, Moles notes that the basic hypotheses of his theory are those of structuralism. Internal organization in a message means a degree of predictability, which is in turn equal to the regularity of the message. Following Wiener, Moles calls this regularity the *autocorrelation function*, which is a widely used way to represent signal properties in signal-processing theory. For Moles, autocorrelation expresses the tendency of any statistical phenomena, including communication, to become structures. Thus a link is established from meaning to correlation and further on to *periodicity*, which can be detected from autocorrelation.

In communication, Moles acknowledges the problem of dynamic structures typical of musical information, among other kinds. Thus, he seeks a more flexible definition of periodicity than that of patterns repeating as exact duplicates, at a constant length known as a period. In his terms, periodicity occurs when we know the coming evolution of a function on the basis of its past evolution. "Knowing" is the key word here, and "periodicity" is taken in a more flexible and relaxed sense than that of being strictly equal. In fact, Moles goes on to say that following periodicity in the classical – i.e., using precise repetitions as the structuring principle of musical communication – deprives the message of deep esthetic interest and results in banality (1966: 69, 150-151). There is a seeming conflict with Moles' concept of information, which is connected to meaning, and the original Shannon and Weaver definition of information, as it concerns engineering; the latter is often cited as a concept having "nothing to do with meaning" (Fiske 1990: 9). Thus, the kind of periodicity Moles is referring to includes both classical periodicity and so-called pseudo-periodicity, which is based more on perceptual likeness than on quantitative similarity.

After establishing the relationship between the internal organization of a message and its signification, Moles curiously turns later to attribute meaning outside the message itself: "Meaning rests on a set of conventions which are a priori common to the receptor and transmitter. Thus it is not transmitted; potentially it preexists the message" (Moles 1966: 197).

There is a possible source of confusion here. It is appears to be a product of the very problem that Morris criticized, namely, the vagueness of the meaning of “meaning”. It seems that “meaning”, for Moles, does not refer to aspects of the signification process but specifically to the base of signification – the conventions, which form the *ground* of a meaningful message. This seeming paradox is further amplified by the fact that Moles does not make a clear distinction between what Meyer calls embodied and designative meanings.

## 2.1.5 Summary of meaning

Kofi Agawu addresses the distinction between embodied and designative meanings by reference to Roman Jakobson’s distinction between *introversive* and *extroversive semiosis*, which roughly corresponds to Meyer’s division between embodied and designative meanings (Agawu 1991: 23). Agawu argues that, in musical signification, introversive semiosis dominates. Thus, he yields to the structuralist tradition, which seems to be particularly fruitful in the pursuit of the roots of musical meaning. The same structuralist attitude underlies the philosophy of the present work, and justifies the study of musical signification by means of computation. If we agree that introversive semiosis dominates in musical signification, then musically justifiable structural groupings will lead us to the foundations of musical meaning.

From the point of view cognitive studies, it is somewhat paradoxical that the structuralist tradition has largely evolved in the context of linguistics, and has strong ties with referential significance and extroversive semiosis (Monelle 1992: 56-58). It goes without saying that introversive semiosis offers a more fruitful ground for structuralist analysis, and by extension, for connectionist models of music, such as the present study proffers.

Several answers can be offered to the question, How can the structural grouping of musical objects be established? Before discussing these answers, let us first establish a threefold point of departure:

1. Many computer models of music rely on essentially non-musical representations. In other words, computer programs process musical material

mostly by non-musical, formal syntax. Music is described in a symbolic way. This means that the description has an arbitrary relationship to music as a physical phenomenon, information, data or system.

2. The formal syntax (in the Chomskyan sense) and its accompanying symbolic description do not necessarily have much in common with the "understanding" process carried out by a human analyst (see Bharucha 1987: 24-25).

3. The mechanisms of musical cognition are presently known only in part. Generally, cognitive processes may display greater complexity than do any of the present theoretical or practical models (Kohonen 1988: 13).

An analytic method having a wide scope and applicable to many kinds of musical styles and situations would bring with it a change in the paradigm of computer-aided study of music. Interest would center on processing strategies for musically meaningful units of information, rather than on the processing of arbitrary symbols that are constrained to be interpreted only as musical ones. To date, the problem of segmentation has been solved in ways that are valid for a single musical situation or small musical corpuses. A more general solution, though harder to find, would merit more scientific interest. Whether or not a common denominator can be found for segmentation of music stored in different formats (e.g., sound files or event-type representations such as MIDI files), remains both an interesting and challenging question. The specifics of different storage formats vary a great deal. With musical cognition as a theoretical starting point, it might be reasonable to suppose that some common processing schemata exist that deal with segmentation in general.

The role of cognitive science, in its search for analytical laws, is to provide a framework of concepts. Analytical procedures may be evaluated by cognitive validity. In a way, this may provide a counterbalance to other courses of action, such as the use of mathematical or linguistic principles in the processing of musical data. Because music is not a mathematical system, it cannot be accessed by mathematics alone. Nor is music entirely a linguistic system, for it cannot adequately meet the formal criteria of the latter, such that the words "music" and "language" would appear synonymous. The cognitivist's task is to provide grounds on which to evaluate the kinds of procedures applicable to music and in what context, be they mathematical, linguistic, or of whatever kind. Thus, cognitive science is charged with exploring previously un-scouted territory. Since there are different schools within cognitive science, the role of cognitive validation should also be questioned. The answer probably depends to some extent on the context: one should be able to choose a research method to deal with the chosen problem, a sort of scholarly “home base”.

Into what scientific “slot” does the present study fit, given its multidisciplinary nature? The answer is not a trivial one. The objective of this study is to develop a model of signification that can clarify the relationship between ideas from signal theory, pattern recognition and structuralist semiotics. This goal is pursued by means of computer simulations using musical data. The starting point is communication and the properties of musical signals. The purpose here is not to implement an imitation of the human mind, even within a limited scope. In that sense, the orientation of this study is not strictly cognitivist. One reason why connectionist models, such as the one used here, are interesting is their cognitive justification in comparison to the more traditional symbolic AI (artificial intelligence) models. It must be emphasized, however, that cognitive models are used here as means, not as ends. This work focuses on the neutral level of communication between sender and receiver. Semiotic concepts are applied to the bare signal; the latter is understood here in the broad sense of any stochastic time-sequence, not as limited to physical quantities such as electrical voltages going to a loudspeaker or other such quantities. The study goes beyond implementation and observation of a pre-described computer model – such activities are closer to mere demonstrations than to fundamental research. Targeting the neutral level justifies computer simulation as a type of empirical study, which aims to increase our understanding of processes that take place at the neutral level, but also helps to explain the receiver’s as well as at the sender’s end of the communication chain. The information processes along the chain are assumed to be more unified than has been shown previously. Thus, methods often used in the to modeling of musical cognition, are applicable here, even though building an artificial music listener/cognizer is not the principal occupation of this study. The focus is on signal processing, on the regularities that govern introversive musical semiosis, and on possible unification of models.

## 2.2 Cognitive models and computer-assisted musicology

### 2.2.1 On theoretical paradigms of computer music

The roots of computation in the study of music can be traced back quite far (see e.g. Kohonen et al. 1991: 229-230). For practical reasons, this study limits the historical perspective of quantitative methods mainly to the period following the late 1940s. In order to see how meaning and significance have been addressed by various scholars since that time, a brief historical survey is needed.

During the 1960s, computational musicology branched off from the general field of musicology (Smoliar 1973: 123-124; Xenakis 1971: 131-134). The new discipline was a child of its time: the adopted theoretical framework was strongly influenced by statistical information theory, as outlined in the late 1940s by Shannon and Weaver (1949). Indeed, statistical methods constituted the leading paradigm of computational musicology during that early period. Attempts to utilize statistical tools for analysis and generation of music evolved quickly soon after the publication of Shannon and Weaver's theory. Yet a significant amount of work in the field took place before the computer became available as a tool for music analysis. Pinkerton (1956: 86), for instance, only mentions the possibility for machine implementation while introducing his experiments on an information-theoretical approach to the generation of melodies.

In the early years of the discipline, musical computation was done just as much with pen and paper as it was with electronic computers. The Greek composer Iannis Xenakis was one of the most dedicated pioneers in the field. Some of his music relies on statistically based calculations carried out by hand (e.g., Xenakis 1971: 83-84) or on decisions made by musicians (ibid: 122-127). Due to the nature of Xenakis's methods, however, he was immediately prepared to use computers when they became available. Among the "hand-crafted" pieces of research on information-theory based modeling of music, David Kraehenbuehl's and Edgar Coons' early publication leaned toward analysis. The authors brought out seminal ideas, and tried to explain the principles of information-theoretical analysis to the uninitiated and to less mathematically inclined music scholars (Kraehenbuehl and Coons 1959: 510-511). At Harvard University, Joel E. Cohen produced an even more complete view of ways to implement information theory oriented procedures in musicology. Cohen (1962: 137-138) also anticipated the influence of general linguistics in musicology, by referring to the syntactic study of music – an idea that gained more ground a decade later. The distinction between information-theoretical/statistical and linguistic/grammar-based modeling of music remained unclear, however. During the 1950s and 60s, information theory also had a strong influence on linguistics. Even Shannon and Weaver referred to linguistic concepts, despite the fact that their primary interest and bases of their thought were in applied mathematics, information physics, and in a very practical interest in engineering problems.

To summarize: it is quite evident that the early steps taken in computer music composition and research were no less driven by theoretical development than by the advent of computer hardware. The first applications of computers to music were born out of demand – the methods adopted by composers and scholars required them.

Noam Chomsky's theory of generative grammars was perhaps the most significant step in the general linguistics of 1950s. As Chomsky's work began to produce offspring in other disciplines, computational musicology gave the new paradigm a warm welcome. Generative grammars were technically feasible, since the theory of formal grammars embraces that of abstract automata. Automata theory became very significant during the computer era, by offering ways of practical testing with modest effort. Even more importantly, formal grammars became an integral part in the development of computer science, since the whole business of giving tasks to computers is largely the science of expressing things in formal languages. In musicology, the language metaphor had long been under discussion, arising partly from the ties already mentioned, between probabilistic models of communication and Chomskyan linguistics.

At the same time, another path led scholars towards linguistic models. The development of semiotics, or *semiology* as it was generally called in Europe, stemmed from linguistic ideas. Semioticians often consider semiotics as a special branch of semiotics. According to Jean-Jacques Nattiez's (1973: 51-57) review of the early development of semiotics, many ideas in the field were born in general linguistics. As he put it, “The idea of using linguistic models for the study of non-linguistic fields... ” goes back to Ferdinand de Saussure’s *Cours de linguistique générale*, first published in 1916. Nattiez further mentions that one reason for importing linguistic-semiotic ideas into other disciplines such as musicology was “to bring rigour to where there had been mere laxity” – i.e., to make the discipline more scientific. These very words also express the spirit that reigned among the early scholars of computational musicology. Little wonder then, that the ties between musicology, linguistics and semiotics strongly manifest themselves in the work of such scholars as the Italian musicologist, Mario Baroni (1983: 175-185).

Much effort has gone into the development of powerful grammatical descriptions of music (see Roads 1985: 419-428). Consequently, the linguistic metaphor constituted the major framework for computational musicology during late 1970s and early 1980s.

Grammar-based computational models prepared the way for artificial intelligence (AI) methods and musical expert systems. Knowledge therein is represented in the form of a rule database that gives an exact description of all knowledge present (Thomas 1985: 268-269; Ebcioglu 1988: 47; Holtzman 1980: 32-34). Thus, expert systems are closely related to musical grammars, scarcely producing new ideas other than the database technology used to manage the knowledge. Roads (1980: 15-16), in discussing historical developments in artificial intelligence, states that *systemic grammar*, as used by Terry Winograd in the 1960s and 70s, paved the way for AI in music. Another early source cited by Roads is Otto Laske’s vision of an intelligent musical robot working “on the basis of musical-grammatical constraints” (Laske 1975: 71).

The late 1980s and early 1990s saw growing interest in massive parallel computation, connectionist machines, and neural networks in musicology – an interest, which followed a general trend in the humanities, and might prove to have been a paradigm shift even greater than that of the turn from statistical to grammar-based systems. The basis of thought in neural and connectionist systems radically differs from the theory of rule sets. Still, there remains a common factor between the two. All the aforementioned architectures have at some time been regarded as model of cognition. Consequently, the development of cognitive science displays a rather intimate relationship with the general theory of information processing.

During the first 30 years of computational musicology, the line of thinking moved from statistical mathematics to general linguistics and further on, to sub-symbolic cognitive modeling. Is the role of computational musicology always going to be to import ideas from other sciences into music research? Can it be regarded as an independent discipline at all? The answer to these questions depends on the following: How well do the methods of computational musicology fit the special needs of music research? To what extent do the imported ideas remain foreign to the study of musical models and musical information processing? What is the objective of computation in musicology? How original are the ideas and algorithms that make a successful musical computer application? To find adequate answers to these questions, one must go deeper into the foundations of musical computing. Let us now try to clarify further the motivation for using computational methods in musicology, as well as level some criticism against it.

### 2.2.2 Cognition as a research topic in musicology

The justification and need for computational musicology have been subjected to a considerable amount of debate. Harry John Maxwell (1984: 2) presents a typical example of this discussion. Maxwell's work deals with a method for harmonic analysis using an artificial intelligence type of computing environment. Maxwell compares his own point of view to that of Allen Forte (1967: 33-34). According to Forte, the main motivations for computer-aided analysis are completeness and precision of analytical method. This is reminiscent of Nattiez's request for methodological rigor, as mentioned above. Forte also refers to the reliability of computational methods. A number of authors after Forte have adopted the same point of view (e.g., Baker 1989: 312; Baroni et al. 1982: 210-211; Broeckx and Landrieu 1972: 32; Frydén and Sundberg 1984: 221; Holtzman 1980: 2-3; Schottstaedt 1989: 199).

Completeness and precision are important qualitative aspects of scientific work, but not analytic goals as such. In deriving a new method, one of the challenges is to make sure that the definition of scientific goals is clear. The challenges of the method crystallize only with careful analysis and application to practice.

Curtis Roads sets the goals for grammatical descriptions of music as follows (1982: 7):

> A main goal of developing more effective representations for music is to improve musician-machine communication, replacing the current rigid protocols and shallow user-interfaces with deeper and richer dialogues. Another goal is more scientific – to develop better models of human musical cognition.

Roads considers the cognitive part of representation as a focal point. To paraphrase, his crucial question is, How faithfully do various representations match our inherent mechanisms of data reception – i.e., human cognition?

Similarly, Lischka (1991: 434-435) considers the cognitive aspect of formalisms of data processing as an important issue, and he identifies non-symbolic ways of data processing as the source of new tools for musicology. Non-symbolic representations have properties that are very interesting in relation to cognitive models. These will be discussed later.

The cognitive aspect has also arisen in connection with statistical models. David Lewin (1968: 51) presents a well-known statistical method, a stochastic process called a *Markov chain* model as an approximation of one who listens to serial music:

> Imagine a listener who "listens two intervals back" at any given moment, and suppose that, over the course of a piece, he develops a sensitivity to anticipating the probability that any given interval will follow the succession of two intervals just heard. To a quite significant extent, he will then be able to perceive the presence of row-structure in the piece, over the long run. If he can "listen three intervals back" and estimate probabilities of the following interval, his perception of row-structure will be extremely "good".

Counter-arguments can be presented to Lewin's idea. It would not be too difficult to invent combinations of twelve-tone rows that easily lead astray an analyst using, say, a third-order stochastic process as a research tool. This may be particularly true in the case of polyphonic music. The fundamental importance of Lewin's statement is that he presents stochastic processes as cognitive models. Lewin is in good company with this opinion. Leonard B. Meyer (1967: 15) writes: "The fact that music, like information, is an instance of a Markoff process has important practical and theoretical ramifications”.

More than 30 years after Meyer made that statement, the limitations of stochastic processes as models have become even more obvious, through the development of more powerful representations. The same is true of any scientific paradigm.

### 2.2.3 Performance models and cognitive models

In research oriented toward algorithm development, it has become common to refer to inadequate machine performance. Typically, in reports one finds something along the lines of “with the next generation of computers our system may, when further developed, serve as a powerful analysis tool”. Such statements usually refer to the shortcomings of the machinery at the moment when the work was carried out, but sometimes they may also be indicative of shortcomings in the work. Hence, any such statements soon become outdated and meaningless. Nevertheless, machine performance can not be ignored in research, and it may be one of the factors used to evaluate the elegance and applicability of the study and its results.

In the computation-oriented study of musical meaning one can presently envision at least two main types of software architecture. First, the *artificial intelligence* branch of computer science strives to perform tasks demanding intelligent behavior, without making assumptions regarding the correlation of biological information processing to performance strategies of the task at hand. One might call this a kind of “black box” attitude, which emphasizes the input and output of the system, and show less concern about the internal structure of the process. It should be underlined that the definition of “tasks demanding intelligent behavior” has gone through significant changes during the last three decades. Some tasks once considered challenging have become trivial. Another line of thought, called *simulation* by Baker (1989: 311-312), has set out to model the functions of real, intelligent organisms. The latter model prefers cognitive validity of research to evaluation of output only. The dualism between these two approaches I shall refer as the difference between *performance models* and *cognitive models*. Bharucha (1991: 84) makes a similar distinction with the terms “artificial networks/human networks”. In my view, the word “human” seems a bit lofty in discussions of machines; hence, Bharucha’s terminology is not adopted here.

Cognitive modeling does not necessarily reject aspects of performance. The crux of the theory of connectionist computing systems is that problem-solving methods are closely related to data representations, and that there are certain methods that specifically require simulation-oriented data representations. For many pattern-oriented computing tasks, particularly, the simulation of an intelligent organism may in fact not be an extra burden, but through more powerful representations offer more efficient problem-solving strategies.

## 2.3 Critique against computer-assisted musicology

### 2.3.1 On the probabilistic approach and information theory

The main objection to the use of statistical methods in musicology may be found in the work of Cohen (1962: 137-138), who draws a rather direct parallel between statistical information theory and general linguistic concepts. As Cohen puts it, “In information theory, the output of any information source such as communicator *A* is considered as a stochastic process, i.e., a random source emitting signs according to probabilities” (ibid.: 140-141). The feasibility of using statistical model depends on the phenomenon to be modeled: a musical score, events written in the score, the composer’s mind and intentions, a performance of a musical work measured in psychological or physical terms, and so on. Cohen seems to consider musical scores as the primary targets of investigation, but is also interested in the study of performance situations. The latter may be prompted by many applications of information theory based on verbal/aural communication situations. At the general level, Cohen keeps an open mind to both physically and psychologically oriented measures of information transmission, though obviously preferring psychological terms to merely physical ones.

The problem with Cohen's approach is that it ignores anything that happens inside of the information source. The source is seen as a "black box" producing output. The musical object is seen as a system with a limited number of definite states. It may not be quite accurate to say that the laws governing this output device are completely disregarded by Cohen. Rather, the difference is one between causal and chance operations. The causal laws that govern the internal mechanism of the black box are modeled by a non-deterministic model of transition probabilities between the states of the system. In light of current scientific ideas regarding cognition, this "black box attitude" appears too much taken for granted.

As cognitive models with a wide range of coverage, probability systems are obviously insufficient: there are many musical cognitive processes that exceed the power of statistical description. A causal listening mode is an important one, at least in western cultures today: *A, therefore B.* Research in transitional probability functions does not seem capable of explaining the tensions and expectations set up by western harmonic and melodic progressions and their respective solutions. Moreover, it is easier to couple them with deterministic rule sets or other vehicles, which constitute a more tightly bound memory and rule base.

The value of probabilistic models was frequently criticised during the emergence of Chomskyan linguistic methodology in music research in the late 1960s and 1970s. Lindblom and Sundberg (1970: 76) present one such critique: "the 'grammaticality' of a well-formed melody cannot be adequately captured in probabilistic terms. It seems clear that a theory of musical structure should not be dependent on chance and our good luck...." The authors quote Wayne Slawson (1968), who states, that in order to generate complex music, "we must resort to rules that can refer to themselves – rules that permit 'self-embedding' ... It has been shown that a Markovian process, no matter how complex, cannot produce such structures". At the time, it was generally considered necessary to promote linguistic ideas and terminology via a critique of the earlier paradigm. The linguistic paradigm was seen as an improvement over statistical modeling.

The use of probabilistic tools in computer music is nevertheless far from obsolete, even today. Some of the thought that went into stochastic music-generation systems during the 1950s and 60s still remains valid and lives in a number of algorithmic composition systems. Some parts of our perception can be accurately approximated by probabilistic means – especially with complex objects (Xenakis 1971: 8-9). Still, statistical information theory has many limitations. One reason these limitations have become known is that stochastic composition and analysis methods have existed long enough to be evaluated in perspective and to face critique.

### 2.3.2 Cognitive justification of linguistics-oriented models

Musical grammars and expert systems represent a further step in modeling musical communication. Instead of considering an information source as a black box engaged in stochastic behavior, these grammars and systems attempt to explain the causal relationships and structures in the information, even to the point of reflecting the internal workings of the source. The representations used in grammars are basically symbolic, and often strongly relate to verbal communication.

Linguistic theories of music are currently in the process of finding their own historical perspective as well as their proper theoretical domain, as well as receiving their share of criticism. As with the information-theoretical approach to music two decades earlier, the late 1980s saw a frequent need to bring up the shortcomings of the previous theoretical framework. Music-linguistic theories are based on the concept of expressing musical entities in terms of grammatical or logical rule systems – an idea introduced before computer technology was available (see, e.g.. Schillinger 1978: 34-40). This basis has given rise to a considerable amount of questioning. Part of this discussion – the part, which is perhaps the least informed – has tried to clarify whether or not music should be regarded as “language” in the everyday sense. Discussing whether or not music is a formal system or language does little good unless it is understood that any statements about the matter are not definitions; they are aids, scaffoldings and models for thought, and are not to be confused with what is real. A thorough account of such debates lies outside our present interest. For a brief overview of the issue, the reader can turn to Gareth Loy’s account

(1991: 26-27). Here, we summarize the principal problems of what he calls "linguo-musicology":

1) The representation of ambiguous data.
2) The justification of strictly linear, left-to-right processing of music.
3) The number of alternative, differing sets of rules defining the same passage of music.
4) Oft-appearing irreversibility in the analysis of a musical passage, or "composing out" the passage again from the analysis results.

Let us address each of these problems. If random numbers are used as a solution to ambiguous situations, the critique leveled against probabilistic models can be directed to rule-based representations; namely, that such systems generate structures "dependent on chance and our good luck" (Lindblom and Sundberg 1970: 76).

In the light of contemporary cognitive science, the causal, sequential representation of musical communication and musical sign system looks overly simplistic, in much the same way as do probabilistic representations (Clarke 1989: 11). Grammatical systems have been built, which can handle several points of view on the same situation, as well as on ambiguous situations (Winograd 1968: 9-10; Roads 1985: 415-419). Nevertheless, the inherent nature of a grammatical representation is mainly that of a symbol. Though symbolic representations were once regarded as cognitive justification *per se*, since the late 1980s an opinion has prevailed among many scholars (e.g., Lischka 1991: 433-436), that symbol-manipulation based rule systems do not possess sufficient cognitive justification in music. Marc Leman (1992: 43-44) is a bit loose in referring to "real dimensions" of music, apparently suggesting that musical communication can not be distilled to a few scalar parameters, as happens in printed scores of western music: "The main criticism is concerned with the fact that the musical objects with which the linguo-musicology is working are considered too abstract, too much restricted to only a few parameters of score-based music, without taking into account the 'real' dimensions of music."

According to Leman, the use of symbolic rule-description in music in general derives from the linguistic metaphor of music. The linguistic metaphor may seem attractive from the computational point of view, but in Leman's opinion it is hard to justify epistemologically. There seems to be no evidence that the "musical black box", or source of information, possesses such internal dynamics as might be represented with a formal rule system. A number of scholars consider this an important point. Otto Laske (1988: 43) has also expressed his concern about the one-sided emphasis on "logico-mathematical and linguistic intelligence" in computer-assisted music research. Laske extends his criticism to artificial intelligence techniques and expert systems: "... the Cartesian notion of intelligence (viz., the separation of knowledge from action and being in the world) that underlies most artificial intelligence research is largely inapplicable to music research" (Laske 1988: 44).

If Laske's critique seems directed mainly against linguistic and grammar-inspired representations of music, it certainly applies to many other descriptions as well, some of which belong to more recent connectionist kind, which has claimed more validity and credibility from the point of view of cognition. If inseparability of knowledge from action is used as a governing principle for music representation, we may come to see that not only generative grammars but also some newer knowledge systems tend to disobey this principle, too.

In connectionist systems that process symbolic data, the foundations of knowledge do not depend on actions of the system, but rather on *pre-programmed knowledge*. Such knowledge is independent of the action. It might be described as a "pie-in-the-sky" type of given knowledge, which the model itself does not explain at all. This "symbolic connectionism" has been severely criticized by Kohonen (1989: 265):

> The distributed models of neural networks can further be divided in two categories, which are based on completely different philosophies. In one of them, also called the "connectionist" models, a very definite and distinct meaning is assigned to each node or neuron in the network. The neural network is thereby conceived as a direct analogy of the abstract *semantic networks* where the nodes correspond to concepts or other linguistic variables, eventually verbs. In order that this be possible at all, it would be necessary to postulate that the role of the biological neurons as operators in semantic interpretation were determined genetically. On the other hand, contrary to a common belief, there exists rather little physiological evidence to such semantic nodes in brain networks. This kind of modeling is also a "brute force"

approach, because it bypasses and ignores the most intriguing question about the automatic formation of such semantic nodes, and does neither pay any attention to the encoding of the inputs, or mechanism for the interpretation of the output activities from such a network.

On the other hand, it is unreasonable to think that the neurally/ biologically based computing metaphor, with its integration of knowledge and action, is a single, neatly unified package of thought. Study of biological mechanisms of data processing has brought about a fair number of proposals for computing architectures that share some common ideas, but differ significantly at the feature level, and even imply contradictory operative principles. Recent novel ideas have come, for instance, from the study of firing rates of neurons, and of the ramifications of the firing oscillation itself actually being a significant information carrier, even extending to a sort of pulse-coded information embedded in the firing signals (see, e.g., Laine 2000: 65-71).

In sum, weaknesses of rule-based musicological models stem from incompatibilities between the cognitive reality and the model, as well as from the incomplete and competing theories about the actual nature of the cognitive reality. Other criticism could be brought forward, of a more immanent nature, concerning the conflicting demands made on the rule system by the objectives of analysis.

### 2.3.3 Complexity, specificity, and meaning

Rule-based music representation systems usually need a way to cope with the dialectic of *generality* and *specificity*. Let us consider an imaginary rule-set that represents a musical work or style. The rules of the set are to be formal, so as to enable computer processing. This generally means that the rules should be written with a particular case in mind. On the other hand, the rules should also represent invariances present in the data, such that they should generalize. They should preclude redundant information. Without an ability to generalize, a covering rule-set is bound to be larger than is practical, and probably very complex. On the other hand, a smaller set, which cannot be applied except in some peculiar situation, is not a very interesting subject of study. On these conditions, we arrive at two results that, applied to any system, work in quite a contradictory way:

1) A highly specific set of rules is irrelevant in the analytic sense, because no reduction is made. Thus, such a set contains little coverage of general musical problems.

2) A rule-set of high coverage is able to make powerful reductions and is applicable to general problems. On the other hand, it tends to resist formalization, and thus is of little use in computer-assisted study.

The simultaneous demands of specificity and coverage are hard to satisfy in a single rule system. A possible solution to this problem is to use a special context-sensitive grammar, which automatically adjusts the length of context of the rules, according to the logical content of the data (Kohonen et al. 1991: 230-242). Even context-sensitive grammars, though successful in certain pattern-recognition applications, do not seem to produce rules at a level of abstraction high enough to access the cognitive mechanisms of musical signification; at least, one might say this upon reviewing examples given in Kohonen et al. (ibid.: 241-242). Limited capability of abstraction seems to be a general property of non-hierarchical formal systems applied to musical data. Such systems can represent little more than the surface level of musical structure. A further conclusion can be drawn: a good representation of music would support knowledge abstraction and reduction, and allow for the storage of information in a hierarchical manner. Moreover, the reduction system would be more elegant, and perhaps even more powerful, if it were cognitively justified.

Even a hierarchical model, like that of Lerdahl and Jackendoff (1983), is not very flexible in the cognitive sense, but seems targeted toward clear and definitive formal solutions (of course, the authors do not claim otherwise). Still, structural uncertainty is common even in the classical repertoire of western music, in the sense that a musical object may belong to several levels of hierarchy. A passage or an individual sound or note object may also belong to one or more *linear* units of form, in Schenkerian and Lerdahl-Jackendoff styles of parsing (see Lerdahl and Jackendoff 1983: 258-259). Mechanisms for dealing with such ambiguous data may be incorporated into a formal rule system, but it makes the representation a bit less charming in terms of "computing aesthetics".

One of the most severe problems of grammar-based and AI music representations is their relationship with musical meaning. Successful rule-based systems suggest a much simpler concept of meaning than can reasonably be thought to describe musical communication. This is probably what Leman (1992: 43-44) probably referring to in stating that "the 'real' dimensions of music" are not taken in account. James R. Meehan (1980: 63-64) points out that:

> AI models for natural-language-processing may work well on ordinary prose such as newspaper stories.... They cannot yet recognize literature because there's no representation of those domains that define literature as something beyond simple prose ... just as there are programs that compose nursery rhymes, there are programs that produce poetry, too, but they're on equally weak foundations.

Meehan's criticism is particularly serious when artistic communication comes into play, for meaning in artistic creation at its best goes far beyond literal meaning in simple texts in complexity, depth, and density.

For practical purposes, however, grammar-based analysis still attracts interest in music studies. An appealing trait of linguistically oriented descriptions of music is that they lend themselves rather naturally to data processing and to descriptions made in programming language. This is so because the behavior of a computing environment is frequently and easily represented as a formal grammar. Moreover, as Roads (1985: 405) states, it is wrong to consider all linguistic-oriented research as a unified whole, for a massive body of various models have arisen since the late 1950s, such that limitations discovered in one model might not be present in all of them. As with probabilistic data representations, linguistic modeling, too, is closely related to computing methodology, which was developed along with data representation. In a broad sense, however, linguistic orientation is more an *attitude*, or manner of viewing things, than a certain algorithmic implementation or computing strategy, or even a group of strategies. One might assume that such an attitude would adjust to many kinds of systems, just as the notion of grammar is used in a broader or narrower senses in various contexts. It is an attitude that leans toward cognition and that is exhibited in the use of linguistic models – an attitude in favor of a *symbolic theory of musical cognition.* As far as computing strategies are concerned, then, it is incorrect to say that the days of the linguistic attitude are over in music studies. Such an attitude may serve well in the construction of data processing models of music, as long as its role and limitations are understood.

### 2.3.4 History, culture, and human reception

*Connectionist,* or neural network-inspired, representations may well overcome some of the problems encountered in linguistic and probabilistic data models. They approach and approximate current knowledge of data representations in biological organisms in a flexible and adaptive manner. Still, connectionist representations are not without problems, and have drawn criticism – a kind of criticism that in fact can be extended to all attempts to develop cognitive data representations and models of information processes.

The cognitive approach to music studies is sometimes accused of being completely ignorant as regards the history of musical idioms. Computer-assisted musicology often faces the same criticism. It is true that such studies focus on musical works or bodies of works, mostly of the same period, and leave historical developments untouched. Such studies search for either musical or cognitive universals, which leaves little room for engaging processes of diachronic development. Yet cognitive research should be able to provide explanations for music of various historical periods. The results of cognition-oriented study are also to be questioned and evaluated in the light of history. The questions to ask might include the following: Do factors exist presently that might at one time were considered universals but are no longer thought of as such? What historical developments have taken place in the relation between music and these universals? Such questions seem to have received little attention in musicology. Xenakis, who argues that mathematics should also be seen as a “cognitive model” of a certain kind (!), has presented a diagram of the parallelisms in the historical development of musical and mathematical thinking (1985: 188-191).

Any cognitive model of musical communication is challenged by the great variety of musical cultures and idioms. According to Dowling and Harwood (1986: 4), the diversity of these cultures implies that musical cognition strongly depends on information processes that are both flexible and context-sensitive. Many of these processes obviously operate on the subconscious level. This implies that the actual musical universals are either relatively small in number, or that they operate on such a fundamental level in the hierarchy of cognitive processes that they allow for highly diverse music-cultural manifestations. Thus, study of musical cognition involves explication of a manifold of musical worlds, whether they are outlined by diachronic (historical) or by synchronic (cultural) separation.

A characteristic feature of the human reception of music is one that might be called *kaleidoscopic reception*. The listener's attention is often divided between a number of things: simultaneous progression of musical form on several hierarchical levels, harmonic ambiguities, more than one melodic line (counterpoint), to name only few. Discussions about the dimensions and data representations of musical communication sometimes confuse representations of music with representations of individual notes in a score or MIDI flow, which can indeed be closed representations, characterized by necessary and sufficient conditions. Music is a multi-faceted, multi-dimensional phenomenon, however, where concepts with necessary and sufficient definitions are often not very useable. Reception of musical communication must be just as multi-faceted and multi-dimensional – open to the point that the number of dimensions and facets may be constantly variable.

Diversion of attention and multiplicity of focus in musical styles, cultures, thought and cognition have remained as hindrances to any unified methodology in computer-assisted music studies, as well as to cognitive theories of music in general. After all, the working mode of a computer is governed by a sequence of serial instructions. For rule-based processing the multi-faceted, kaleidoscopic stream of musical information raises important questions: To what extent is it feasible to implement multi-dimensional or variable-dimensional information in a networked form, with a methodological framework that inherently operates on a linear metaphor? Does it make sense, or is it even possible, to simulate multi-dimensional processes on a data processing platform that operates in a linear manner?

Probabilistic methods of analysis seem slightly better equipped to deal with complex information processes involving multiple foci, because of their ability to extract general information from particulars. A large number of instances, when processed collectively, allow for diversion without sacrificing the global perspective.  The price paid for such an allowance is that a particular object essentially occurs in probabilistic analysis only within its context – its existence as a separate entity is meaningless.

Connectionist research of the 1990s seems to have found a way around the trade-off of particular and general, which results from choosing between probabilistic and grammar-based music representations.  A connective machine, as an analysis tool, may accomplish this, not by making compromises with the older data representations, but by introducing completely new ones. Being more flexible by nature, connectionist representations may incorporate a number of kaleidoscopic viewpoints into one.  This comes only with the caveat that real-life connectionist data processing systems are gross simplifications of true neural processes.  Nor do they run as actual, analog parallel processes, but as digital, serial simulations of such processes.  Even though connectionist systems have been built to imitate parallel processing, chances are that fundamental differences between those representations and real analog parallel processing – even analog electronic ones, to say nothing of biological information processes.  As yet we have no answer to the question of how much better, if any, connectionist models are, at approaching deeper and more complex types of meaning than mere literal meanings at the most elementary level.  After more than a decade of connectionist modeling, the time of describing connectionist research as “a promising new approach” has ended. Hence, it is time to evaluate how far the framework has been able to support music studies.  As a rule, such evaluations are prompted by the next new approach or paradigm shift in the field, which provides refinement of earlier ideas.  At this time, however, a major shift toward the next paradigm is not yet taking place.

## 2.4 Describing musical objects

### 2.4.1 Event and flow as data-representation levels of a musical object

There are at least two different ways to represent musical objects, which could be referred to as *event-oriented* and *flow-oriented*; that is to say, as discrete and continuous, symbolic and physical, or, in the semiotic sense, as predominantly symbolic and predominantly iconic. Our interest is in the needs of data processing, but problems of representation can also be addressed without specific reference to automation. Let us first concern ourselves with the division between symbolic (convention-based) and physical (non-conventional) representation. Physical representation has close ties with the concept of *iconicity* as defined in semiotics, further expounded in the next chapter. In the spirit of cybernetics, one might say that the distinction between symbolic and physical representations resembles that between a physical sequence and the control information implementing it. It is also reminiscent of the division between control data and audio data, which can frequently be found in both digital and analog systems of sound synthesis. Control information represents an upper level of process hierarchy, and facilitates handling of a complex system by reducing the amount of data necessary for control, and by offering a uniform interface. Symbolic representation is a data reduction, allowing the physical representation to be retrieved through a process of interpretation. Thus the original “raw” data may be retrieved at another time or in another place, if its symbolic representation has been preserved. This also resembles the “expanding” of compressed data in many data-packing systems widely used e.g. in the Internet.

Many examples are readily available of symbolic music representations. The most obvious one is probably western musical notation. It is not completely symbolic – a certain amount of similarity is involved – but most people would probably recognize symbolic function as the dominant one in music notation. A good reason for this claim is the fact that music notation engenders *event-based thinking about music*. Among data-processing representations of music, the most usual cases of symbolic ones are high-level definitions of scores.  Score definitions reduce a musical event to a set of attributes, usually scales.  Notation programs such as *Encore*, *Finale*, and *Sibelius* use such definitions, and MIDI sequencers use data structures that are quite similar in principle.  Also sound-yielding programs – acoustical compilers – such as *csound*, *Supercollider* or *MUSIC V*, use data files in their source codes (score and instrument definitions), which belong to this category. The objects to be processed are of the same nature as common-practice musical notation, in the sense that many operations and transformations characteristic of musical notation – and notation-based musical thought – fall within the scope of these representations.  These operations include transformations such as transposition, inversion, retroversion, and intervallic augmentation and diminution.  It is remarkable that these operations, often known as contrapuntal transformation techniques, are often associated with process of composition, which in turn can easily be associated with free, continuous flow of music in a creative and artistic mind.  Somewhat paradoxically, these contrapuntal transformations depend on the idea of music comprised of clear-cut discrete events as building blocks, not as a continuous stream.

Physical representations of sounds involve various forms of digitized sound data.  Perhaps the simplest of these is linear PCM coding of signals in the temporal domain.  This type of coding is symbolic in the sense that numerical coding is used, but not symbolic in terms of discrete, event-based musical thought.  Events are present, in the sense that sound samples digitized at equal time-intervals are directly and linearly mapped to a given range of values.  But an event of this type is not a musical event at all.  Any one such event is just a scalar value, having no interpretation at all with respect to musical meaning – this is an extreme case of a single event receiving its meaning only in the context, and having no meaning at all without it.  It would be naïve to call this kind of signifying process “symbolic”, since the discrete symbolic nature serves only as a basis for the flow of continuous information at a higher level, which is the actual object of representation.

Representations of sound according to frequency domain play role similar temporal-domain representations with respect to musical meaning. They incorporate information retrieved from time-domain signals via a mathematical transform (e.g., the Fourier transform).  They are essentially spectral representations serving many signal-processing computer applications, such as sonogram/spectrogram analysis and re-synthesis tools.  An example of this type of device is *AudioSculpt*, a spectral filtering tool used in IRCAM signal-processing software designed for Apple's Macintosh computers. Spectral representations deal with sound signals instead of notes or other musical objects. Like time-domain signal representations, they are symbolic only in a superficial sense. Most commonly used symbolic operations of a musical nature are not directly available for time-domain signals, or require computation-intensive operations in order to be implemented. They differ from time-domain signals in that every time-window is represented by a vector or group of vectors, allowing for multi-dimensional data and thus giving some room – if just a little – for meaning on its own. Still, the position taken here is that it is erroneous to consider such data elements as symbols. They may be symbols in the sense of low-level technical representation, but certainly not in the cognitive sense.

### 2.4.2 Score and music

It has become customary to speak about representing music in computers when the subject of the representation is actually musical notation, printed music – a score. A score is not music, but its symbolic representation by a written medium. Similarly, a note is not a unit of music, but a typographic unit, a grapheme. Between a note and its acoustic representamen there exists a complex process of interpretation. Similarly, a complex process of cognition takes place at the receiving end of the musical chain of communication, between the acoustical phenomenon and the sign it produces in the consciousness of the receiver. This difference is so self-evident that it has sometimes been neglected in music analysis. Computer-aided music analysis in particular is often regarded as score analysis only, dominated by graphically represented dimensions of music description. A number of important qualities, many of them connected with timbre, make only indirect appearance in this type of analysis. Yet timbre is a profound element of any musical phrase. The neglect of important elements is a shortcoming that can not be overlooked, no matter what the analytical framework is. The importance of indirectly marked elements of music is strong in contemporary music, even though tradition has not laid down irrevocable rules concerning performance practices. But it is particularly true of genres, where tradition remains strong. Among others, musicologist and semiotician Eero Tarasti, even in the context of careful score-study, has underlined the importance of properties – such as timbre – that are not directly visible in the score. Giving an example of the English horn in Sibelius's *Swan of Tuonela*, he maintains that in "Western music almost every instrument and its timbre has its own characteristic 'nature' which a composer must appropriately incorporate into the musical-psychological whole of the composition" (Tarasti 1978: 78-79).

In regard to the differences between music and the score representing it, one must note that various scores have various levels of abstraction, iconicity, and symbolism. For instance, tablatures quite obviously belong to a lesser level of abstraction than do scores of traditional notation, because they have a more iconic relationship with the physical gestures of a musician. With MIDI files, the level of abstraction and relationship to musical meaning are somewhat ambiguous. A MIDI file may represent either a gestural score or traditional notation. For instance, a file recorded from the key movements of a MIDI piano may be taken as a gestural score, whereas a file produced by step-editing a written score is in terms of abstraction no different from the original, written score. Thus MIDI information may communicate on an auxiliary level as compared to the score – a series of tactile actions. In quantitative analysis, a gesture-score may be treated with similar procedures as a score of printed music. In addition to score analysis, computer-assisted musicology could also help in the development of analytical tools geared to accessing the tactile aspects of recorded music.

Sound-signal representations introduce problems unlike those encountered in score-representation. Their low level of abstraction and high density of data place practical demands on the data-processing architecture used. One could say that digital sound signals are a minimally symbolic way of representing musical information by computer. Because of their directly physical nature, they tend to preserve an image of musical communication faithfully and completely. Paradoxically, because of the minimal reduction used, they also tend to be quite remote from musical concepts. So precision of reproduction does not necessarily imply analytic potential. *Reduction* of information is also required.

Digital representation always involves reduction of some kind. Some information will always be reduced out, due to the nature of digital coding; the coding itself being a discrete numerical approximation of a continuous, un-quantized physical world. In coding, the omitted information is considered unnecessary, but the judgement of what is necessary and what is not can always be disputed. Naturally, a fundamental part of the preliminary work for any analysis is to provide a definition of the kind of information that has been left out of the analytical procedure, as well as to determine whether or not the reduction has a pronounced effect on the results of the analysis.

For instance, timbre-related properties of sound seem to escape simple definitions. In acoustical literature, the concept of timbre is usually coupled with information concerning the spectrum of sound, its formants, and envelope curves (Backus 1969: 94). Nevertheless, one of the pioneers of the *musique concrète*, Pierre Schaeffer (1966: 55-56), presented the concept of timbre as a part of the sound-color of an acoustic instrument. For Schaeffer, timbre primarily meant the native, invariant properties of an instrumental sound-color. Among these properties is the “violin-likeness” of the sound of a violin; but different tactile and technical features of the sound of a violin, such as legato, pizzicato, spiccato and register, lie at the outskirts of Schaeffer’s concept of timbre. Undoubtedly, the latter properties have a remarkable influence on the entities used for physical definition of a sound, such as spectra and envelopes. Hence, the series of partials (harmonics), spectral structures, and any other physically measurable entities are something other than timbre as it conceived by Schaeffer.

### 2.4.3 Score representations

A number of representations have been developed for musical scores. For the most part, these are incompatible with each other. The prevailing Babel of practices illustrates the general difficulty of building new standards. The basic setting of the problem is simple. There is an object – usually a note or a MIDI event – that has a number of attributes. The number of attributes to be defined for the object, and the properties they should cover, is a source of constant disagreement. Moreover, the ways of grouping these objects into lines, tracks, phrases, riffs, or rhythmic patterns are arbitrary. The same musical passage can appear quite differently in different encoding systems. For instance, attribute “stem upward” would be meaningful in traditional printed music notation, but it is meaningless in a MIDI play-list or in audible music. A standard way of addressing these problems has not yet gained general acceptance, and it seems likely that the ongoing dispute will not end anytime soon.

MIDI, as a global, data-communication specification for musical devices, has promoted the simplistic representation of music. Though suitable for limited purposes only, because of its lack of detail, it has greatly facilitated the storing and processing of musical data. MIDI has made it easier than ever before to collect musical information to serve as test material. The standard way of data storage has also promoted conversion routines for other, more sophisticated representations. This has brought a partial solution to an earlier problem in the development of computer processing of music: too little source material in computer-readable format. On the other hand, reliance on large bodies of data using greatly reduced representation may lead to a misplaced trust in the results, which might be misleading due to a disregard for the consequences of data reduction.

### 2.4.4 Symbolic musical abstractions

In addition to physical and score representations, another class of representations can be defined, which operate at a higher level of abstraction than do the former ones. A large part of the methodology in computer-assisted musicology, such as grammatical models and statistical models, has been that of representation, closely intertwined with the methodology of reduction. The results of such analytical or reductionist procedures can be considered as forming a class of their own. A proper name for such descriptions might be *symbolic musical abstractions*, and they involve discussion of the nature of musical meaning. Roads (1982: 7) refers to this discussion as follows: "... Music representations refer to the formal symbolic specification used within computers for capturing musical structure and meaning."

Computer-assisted musicology displays the traits of a science in a pre-paradigmatic state, since professionals in the field can not seem to agree on such basic questions as representational methodology. Is this something to be concerned about? The choice of a particular descriptive method of a higher level of abstraction often means commitment to a certain cognitive model.

Cleaving to one model also means favoring a particular cognitive theory. Laske (1988: 43-44) points out that the disagreements about basic methodology is due to the interdisciplinary nature of musicology in general. He goes on to present his own suggestion:

> ...What this science [lacks] is a core set of methods shared by all inquiries, as well as adequate tools for testing hypotheses. I submit that cognitive musicology provides such a core set, based on the notion of *knowledge representation*. Although controversial and far from monolithic, this notion is capable of establishing a common base for widely divergent hypotheses regarding musical knowledge. (Ibid.: 44)

A key question concerning symbolic musical abstractions is whether or not they can be arrived at automatically; i.e., if known methods of computation can be used in forming or bringing out the higher-level information embedded in them. This question is compelling from the point of view of methodological development. If it is possible to achieve a musical abstraction by computational means, then the abstraction process will require that the musical object represented be independent of interpretation by human individuals, at least to a quite significant degree. A basic assumption of the present study is that the musical unit or object exists independently of the subjective experiences of the listener, at least to some extent, and within the scope of quantitative description. The musical object also exists independently of scores, of instruments, and of recordings of any kind. The object itself lives in a dialectical relationship with all these entities, but primarily depends on information structures embedded in musical communication.

### 2.4.5 Connectionist representations

Connectionist representations are often contrasted with rule-based, grammatical descriptions of music. Connectionist representations have commonly resulted from a search for more flexible and powerful descriptions than rule-based systems allow (Lischka 1987: 190-191). Connectionism is a paradigm that emerged in the 1980s in computer science, psychology, pattern recognition, and artificial intelligence, and it relies on computation carried out in a parallel, collective fashion by a cluster of “cells”. The basic idea of massive, parallel-data processing has existed for several decades, approximately as long as the mainstream of artificial intelligence (Bharucha and Olney 1989: 341). The difference between connectionist representation and mainstream AI methodology is that the former contains a built-in mechanism which promotes a holistic view of the object. In music, this might mean that an entire piece interacts with its parts just as the parts interact with each other. Connectionist time-series analysis can be thought of as a search for a representation of such interactive forces.

Connectionism is often associated with artificial neural networks, massive parallel processing, or sub-symbolic information processing, and is based on a number of simple, parallel operational elements grouped in the form of a network with interacting connections. Characteristically, these network systems have some similarity to information processing performed by the human brain and by other architectures that process biological data. Each of the various terms used for connectionist systems has its own points of emphasis, and biological counterparts are imitated with various degrees of fidelity. The systems that mainly seek to mimic biological neural functions are usually called “artificial neural networks”. Networks that interpret the idea of parallel-distributed computation more loosely are known as “connectionist networks”. The term “sub-symbolic” emphasizes the emergence of abstractions in a connectionist network, and is often connected with the concept of *self-organization*, which will be elaborated in Chapter 4. In order to avoid confusion, we shall use the term “connectionism” as a general super-class comprised of these different “brands” of brain-inspired computing machinery, though it must be noted that not every expert in the field would support this categorization.

The elements of a connectionist network act simultaneously as both processors and memory units. Each node of the network performs simple computational functions. The network’s ability to record information is based on variable strengths of connections between its nodes, and on variable states of those nodes. The network’s transfer function is brought about by its collective behavior.

Biological information-processing models have inspired connectionism from the beginning. As may be evident from what has been said thus far, the division between performance models and cognitive models also exists in the field of massive parallel computation. This division is sometimes expressed by calling the former a *connectionist network*, and the latter an *artificial neural network*. Still, connectionism in general claims to be closer to physiological models than is (mainstream) symbolic artificial intelligence. Thus, it is often argued that connectionism works better than other systems do, on the grounds that it has better cognitive justification (Bharucha 1991: 84).

The network paradigm has a special relationship to grammar-based logic. Bharucha and Olney (1989: 343) formulate the nature of this relationship as follows:

> Although the behavior of a neural net can be formally described by the rules of a grammar, these rules are not explicitly represented in the network. Rule-based theories that are offered as formal descriptions are vital contributions to an understanding of cognition because they specify constraints that any theory of the underlying processes must meet. However, rule-based theories that go beyond formal description to claim that rules are explicitly represented, as in a computer program, and that cognitive processes are symbolic operations performed on these representations, are fundamentally different from neural nets in their concept of cognition.

A number of different artificial neural network and connectionist architectures have been created for different applications. A common denominator among them is the distributed representation of data over nodes of the network, and connections between the nodes. The distribution of data provides the most interesting properties of these nets – flexibility, adaptability, and the ability to generalize knowledge (see Alexander and Morton 1990: 19).

### 2.4.6 Sub-symbolic and symbolic data

The notion of *sub-symbolism* is crucial to the conceptual apparatus for the description of connectionist systems. Sub-symbolic data processing, like connectionism, is based on the brain metaphor of cognitive processes. Leman (1991a: 7) describes the sub-symbolic level of musical cognition as a representation located between acoustic and symbolic levels of musical data.[2] His main idea is that the roots of knowledge extend into this low, sub-symbolic level. Symbolic metaphor, which is commonly thought to dominate the highest levels of cognitive processes, is built on the foundation of sub-symbols. In Leman's thought, sub-symbolic knowledge represents the more or less physically measurable "raw data", and is directly retrieved from input data that is fed into a connectionist system. The existence of sub-symbolic data gives rise to the organization of knowledge, which may either be a result of a purposive organizational process driven by explicit directions, or emerge as a result of a data-driven process, in which algorithmic direction does not play a major role in knowledge retrieval.

From the definition of sub-symbolic knowledge it is clear that suitable data-processing methods for such an approach are quite different from those engineered for retrieving symbolic information. The knowledge input may be qualitatively different, because the data being processed need not be formalized at the symbolic level. This enables one to better deal with fuzzy and ill-defined information. A set of examples may be used as the source of knowledge even better than prefabricated formalisms can be. Of course, also symbolic systems can use examples as a source of knowledge in, too, but in that case the knowledge would be more strictly limited to a particular case. Sub-symbolic representations, in turn, allow for generalization of a particular case. This is to give support to a previous, implicitly given argument; namely, that the methods, the mechanisms, and the power of expression in computing rely heavily on the data-representation used.

Sub-symbolic information about a musical object as such will provide few analytic results. Because of its low level of abstraction, it does not readily serve explanatory models. Rather, it provides structuring mechanisms for knowledge, which may be of benefit in analysis. Given information, sub-symbolic processing produces structures that make it possible to retrieve knowledge that reflects a higher level of abstraction than do the original sub-symbols. This structuring may take place either through processes previously learned under supervision, or by processes of self-organization. A self-

2 This is Leman's implementation-specific notion only. One can conceive of musical applications of sub-symbolic representations that have little to do with acoustical information.

organizing process may possess knowledge previously learned in an unsupervised mode, or may build its knowledge “on the spot” – in real time, from the given data. Such processes generate symbolic descriptions, which can be presented as the analytical result. They bring out the hierarchical structure in data, if such a structure exists.

Because the self-organizing processes can be made to build hierarchies, it is naturally possible to build process chains that are equivalent to the emergence of multi-layered hierarchies. The symbols emerging in one organization process can be used as sub-symbols on the next level of hierarchy. Jamshed J. Bharucha (1991: 93-97) of Dartmouth College introduces a model built on this principle. His model uses consecutive layers of spectral representation, pitch height, pitch class, pitch class clusters (i.e., concepts of chords and harmonic intervals), and tonal centers leading to representation, which Bharucha calls “pitch invariant representation”. The two lowest layers of the model are “presupposed”, in his terms, apparently meaning that the actual data-input stage of the model is the third layer – pitch class – upon which the upper layers are built. Thus Bharucha’s model, called MUSACT, serves as a scheme of cognition, including mechanisms for organization both in pitch and in harmony (pitch invariant representation) and in producing representations of musical time (sequential memory).

### 2.4.7 Examples of connectionist applications in music research

A number of compositional programs have been built according to the connectionist paradigm. One of these was introduced by Peter Todd, who advanced an “algorithmic composition” scheme based on connectionist architecture. Todd’s connectionist composition machine consists of a well-known supervised learning system called *back-propagation* network, which is employed to learn musical patterns, and to generate new patterns with the knowledge thereby obtained. He considers his method a great improvement over rule-based composition engines: “These methods contrast greatly with the majority of older schemes that simply follow a previously assembled set of compositional rules, resulting in brittle systems typically unable to appropriately handle unexpected musical situations" (Todd 1989: 27).

As concerns analysis, connectionist ideas have so far been applied mostly to non-temporal, "outside-of-time" musical structures, such as vertical harmony, key relationships, and formation of pitch (Laden and Keefe 1991; Leman 1991b; Sano and Jenkins 1991). One analytical work of this kind is Marc Leman's theory of *tone contexts*. He applies the so-called *self-organizing feature map* (Kohonen 1989: 119-157) to constructing a cognitive theory of key relations in tonal harmony. Leman (1991b) gives an example of symbolic knowledge resulting from sub-symbolic processing. He examines the emergence of harmony and tonality vertically, with no reference to temporal structures of music. On his view, certain cognitive structures, such as the circle of fifths, derive more or less directly from combining certain laws of acoustics and mathematics with others from information science. Briefly described, Leman's experiment was as follows: He fed information about the occurrences of various pitch-classes in the piece to the self-organizing map. In his system, both the pitch-classes of fundamental frequencies and a small number of harmonic overtone components were taken into account. As a result of the self-organizing process, a structure equivalent to the circle of fifths evolved on the self-organizing map. Of course, it has long been known that tonal laws and the circle of fifths have a close relationship with the physics of sound. Leman's work, however, contributes to the formalization of this relationship in a fundamental way, by showing what type of information processes can generate this phenomenon, and what side of the phenomenon is purely physical, thus clarifying the role and function of low-level cognitive processes in our reception of musical sound.

Another system, introduced by Robert Gjerdingen (1991), outputs a categorization of given musical passages. Gjerdingen uses the so-called ART system, which is based on Grossberg's *adaptive resonance theory* (Grossberg 1976; Carpenter and Grossberg 1987). On this principle, Gjerdingen builds an analysis schema called *L'ART pour l'art*. ART is an architecture somewhat more complex than Leman's self-organizing map. Gjerdingen promotes ART as a system that can "successfully categorize an arbitrary large and complex domain of analog patterns" (Gjerdingen 1991: 146). By *analog patterns,* he seems to mean the kind of data which, in this study and in general semiotics, might better be called *iconic data*. To judge from his results, *L'ART pour l'art* seems very successful in categorization (Gjerdingen 1991: 146-149). Thus, we have reason to believe that either the principle of operation or the implementation (or both) is at present one of the most successful musical connectionist analytical systems.

One interesting feature of ART is that it considers methods related to signal-processing theory in a very physically oriented manner, and employs them in processing abstractions. This kind of abstraction in question is a signal, if taken as such, but not an exogenous one. Rather, it is a signal immanent in the cognitive system, not something that would directly and iconically represent the physical world outside of the artificial “cognizer”. Hence ART possesses internal representations easily viewed as signals, a feature that is shared by the model developed in this study.

### 2.4.8 Connectionism, reduction, and the science of chaos

In designing analysis systems it is beneficial to know as much as possible about the behavior of data sources, and this is closely connected to their internal mechanics and structure. If we instead take a “black box attitude” concerning the signal source, the motivation for that choice would be the presupposition that knowing the source’s internal structure is not necessary for analysis. However, there are ways to approach the dynamics of a signal source on a general level without knowing its structure in detail. One such way would connect to the concept of *chaos*.

What we know about chaos might contribute in a way to what we know about connectionism. Research on chaotic phenomena has taught us that order and chaos coexist in many dynamical systems, and that certain features of chaotic behaviour actually reflect regular properties of an information source, or at least give us information concerning the source. This is particularly valuable in situations when the source is complex in structure, which would present obstacles to studying it in detail. Inspired by findings of chaos theory, special techniques have been developed for data compression, and thus are related to analysis and abstraction. One such technique is called *IFS* (*Iterated Function Systems*) compression. An IFS compression, though computationally effective and elegant, has so far offered little explication of, for instance, musical concepts. This is not due to practical reasons, such as computational power, for it probably is quite rich in such respects. The low explanatory power is more due to the fact that IFS has not been bridged to musical cognition in a generally applicable way. Nevertheless, connectionist networks are dynamical systems, some of which are quite complex ones. If musical cognition can be studied with the aid of connectionist networks, it is possible that networks, in turn, can be studied with the aid of tools related to chaos theory. In this way, what we know about the special apparatus designed to control and understand chaotic phenomena could be linked to what we know about cognition.

Another link between chaos studies and cognition might be more direct research into the micro-level of sound. Thus far, we have been most interested in the orderly aspects of musical signals. Chaos science may serve to explain *disorderly* behavior at the lowest levels of musical events: sound signal waveforms and spectra. With many natural (mechanical) oscillation processes, the wealth of various time-variant fluctuations and disturbances may be understood as the behavior of a chaotic attractor system.

Such disturbances form one argument for chaos-based explication of musical signals. For instance, they make the sound of acoustical instruments rich and lively in comparison with synthetic sound. Richness of behaviour is often linked to feedback lines embedded in the oscillating information-source. Whenever energy is brought to a feedback system, a possibility of chaotic behavior exists.

Because our purpose is not to use connectionist methodology as an objective *per se*, it seems appropriate to make a distinction. On the basis of what we know about signals, it is likely that the micro-organization of musical signals is more open to explication in terms of nonlinear feedback systems, whereas higher levels call for explanations with connectionist theories. In any case, the chaotic nature of certain connectionist phenomena may be significant to cognitive research, too.

## 2.5 Modeling music and cognition: Summary

By focusing attention on quantitative aspects of musical meaning, and doing so in as general a way as possible, by relying on concepts of information transmission and message, we have arrived at a structuralist kind of analysis of abstract form. This means turning away from intertextual and designative meanings, and toward immanent, or embodied, meaning. As a result, we turn to introversive instead of extroversive semiosis. Moreover, pursuit of introversive, abstract form has led us to search for cognitive validation when evaluating certain models of data representation and data processing. The reason for this comes from the fact that introversive semiosis is likely to be situated at lower levels of human consciousness than is extroversive semiosis, in which case, questions about cognitive validity cannot be overlooked. We now take a moment to consider the nature of cognitive validity.

It has become evident that no single method of representation in musical computing can be called *the* one and only cognitive model. We have adduced cognitive aspects of probabilistic models, generative grammars, abstract automata, and various sub-symbolic data representations. Even mathematical models, such as those of Xenakis, and representations connected with chaos theory have been viewed as cognitive models of music. As our knowledge of human reception of music increases, current sub-symbolic representations will undoubtedly be replaced or augmented by other schemas, which may claim their share of cognitive validity, or a better approximation of it. Now we arrive at the principal result of this chapter – cognitive justification – which can be formulated as follows:

**Definition 2.1:** Cognitive justification in musical computing is not an irrevocable attribute of a certain model or representation, but an *attitude* by which we approach any given model or representation.

This attitude is a way of using our model such that cognitive aspects make up a significant part of its *raison d'être*, and must be taken into account, at least to some extent. The model must not be misinterpreted as reality, but considered as a tool with which to approach reality. No quantitative model of music can fully share the complexity of the musical mind that it is reflecting.

# 3. Introduction to semiotic systems and categories

## 3.1 Communication, signs and signals

Musical communication, as any other form of communication, is based on a sign system. The sign systems for music, which are only some among the countless sign systems in our everyday environment, serve as alphabets for each of the languages upon which the communicative function of music is based. Signs surround us every day of our lives, no matter where we go or what we do. Interest in signs is the key element in many different kinds of human activities, both business and pleasure, both professional and amateur – more or less in all aspects of life. Our everyday encounters with different sign systems extend from social norms of behavior to highway codes and far beyond.

Scholarly interest in formalisms of signs and communication dates extremely far back in human history. In his book *Digital Mantras*, Steven Holtzman offers interesting insights into the beginnings of language systems. According to him, the first systematic and comprehensive studies of linguistic communication known to date is likely the one from the Aryan culture in Mohenjo Daro of North India around 2000 B.C. Aryan priests wielded tremendous power over the people. According to their teaching, it was absolutely vital that the original forms of Vedic hymns be preserved intact from generation to generation. Preservation of the language over long periods of time was essential for this purpose. Consequently, the priests took upon themselves the task of constructing a grammar of the Sanskrit language of the Vedas, to such a degree of formalization that it could assist in preservation of highly sacred religious texts. This was the motivation for constructing the earliest known generative grammars (Holtzman 1995: 7-14).

Interest in musical sign systems is not a newcomer to communication studies, either. It seems to date approximately as far back in history as do studies motivated by language and religion. Ancient indeed is the idea that music has remarkable power to convey extra-musical messages; the roots of this notion can probably be found as far back as one can trace historical sources. The earliest recorded examples of reasoning about music's ability to carry meaning date from antiquity. In a treatise from the early 6th century A.D., Boethius attributes to Plato a concern about the moral effects of unworthy music – a concern has since surfaced frequently in numerous forms up to the present time:

> …there is no greater ruin in morals in a republic than the gradual perversion of chaste and temperate music, for the minds of those listening at first acquiesce. Then they gradually submit, preserving no trace of honesty or justice – whether lascivious modes bring something immodest into the disposition of the people or rougher ones implant something warlike and savage. (Boethius: *Fundamentals of Music*, chapter 1)

Scientific and philosophic interest in sign systems as the base of communication have resulted in more extensive work on the matter in categorizations of signs, and have triggered studies concerning relationships between structure and meaning in messages. Such studies gave birth to semiotics as a formal study of sign systems and the emergence of meaning. The process by which meaning rises in sign systems has been called *semiosis*. Thus semiotics, or the formal study of sign systems, has developed mainly as a philosophy having communication and general linguistics as fundamental ingredients.

After the appearance of groundbreaking works in semiotics in the late 19th and early 20th century, new disciplines have arisen that also deal with sign systems and communication. Called for by the rise of modern electronic applications of communication technology and initiated by Claude Shannon's and Warren Weaver's classical work, *information theory*, from the late 1940's on, was a predecessor of widespread interest in communication as seen from the point of view of engineering. In my own view, information theory has much more to say about *signals* than about signs. Signal here would be defined as a fluctuating physical quantity, and variations in quantity would represent coded information. Such fluctuations, of course, are bound to time. In typical examples of semiotic concepts, archetypal signs are not often time-related, though there seems to be no particular reason or need for this. Signs and signification processes can be time-dependent as well as time-independent. It might, however, be conceptually easier to operate on time-independent signs, which is perhaps why we repeatedly do so.

As a result, when using engineering terms to address matters of communication, it is customary to talk about signals instead of signs. If in semiotic discourse we replace the word *sign* with the word *signal*, it would enable us to open a discussion about a number of technical tools and apparatuses, perhaps enabling us to gain new insights into communication and semiosis. In the engineering sense of the word, a signal is time-dependent by definition. Considering musical signs – our main subject – it is quite evident that many of the are inherently time-dependent, which naturally leads to a discussion of signals, too. Before that, however, we should first consider some basic ideas about sign categories.

## 3.2 Message and form: Particular and general

It has become commonplace to say that form and meaning in music are intimately connected, as compared to other forms of communication, and especially as compared to other forms of art. But what exactly do we mean when we say that musical communication is form-dependent in a "special" way? Surely the principal meaning of a musical work is not the archetypal, abstract models of large-scale constructs such as fugue and sonata form, or of smaller ones, like the 12-bar blues formula. Instead, it seems much more likely that the essence of a musical message lies in the unique artistic solutions to problems of form, as applied in particular musical situations.

Thus, to claim that an intimate relationship exists between musical form and musical meaning, we first have to distinguish between general and particular, between an archetype and a particular case. One-of-a-kind situations dominate in the formation of meaning and transmission of messages in Western music, despite the latter's heavy reliance on tradition and culture. In the European way of thinking, the creation of music could be seen as threefold. One task would be to expand one's knowledge and mastery of the cultural tradition, another to concentrate on a specific problem in a musical situation, and yet another would be to develop unique solutions for the problem at hand, based on the knowledge of tradition. In essence, this view of the creative process can be seen as a musical case study. "Case study" is to be taken here in more of a synthetic sense than an analytic one; that is to say, more as producing cases of output that retain their link with tradition, while at the same time adding to that same tradition so as to keep it alive and evolving.

This is not to say that archetypes would play no role at all in the transmission of musical message. For instance, in Hindustani music the theoretical archetypes are both numerous and very strong, to the extent of being able to communicate extra-musical messages. This seems mostly due to the complexity of the Hindustani musical system, which in some respects by far exceeds its European counterpart in expressive power. But even in a theoretical structure so rich in the representation of emotions as Hindustani music is, musical expression is much more than a concatenation of numerous elements picked from a ready-made list:

> Just as the taste of a delicacy is not merely the sum-total of the taste of the ingredients, but is something quite new wherein the ingredients cannot be perceived separately, similarly the content of enjoyment of art is not the sum-total of the various components of artistic representation, but is quite different. (Sharma 1970: 58; quoted in Martinez 1997: 199)

Thus, even in systems in which the role of arbitrarily assigned extra-musical meaning is very strong, the importance of synthetic perception remains highly pronounced. Reception of music is evidently a situation in which one plus one equals a good deal more than two.

On the other hand, in some cultures a musical archetype – the structural scaffolding that comes from tradition – is so strong that it has almost complete control of the large-scale form. For instance, the role of verse/chorus in much jazz is strong enough to enable rich communication in message, expression, and taste without contradicting listeners' expectations concerning the scaffolding in any major way. As an even more extreme a case, the repetitive loops in Steve Reich's minimalist works might seem to generalize the formal aspect nearly *ad absurdum*. Yet amid all the repetition there still is room for a small, unique feature here and there to be woven among the threads of the slowly and steadily evolving musical fabric.

A question worth asking, in regard to music studies, is if the mechanisms of musical meaning are peculiar only to musical communication, or if more general rules governing meaning can be found. In studying various modes of communication, it does not take long to discover that, some features are mostly present only in musical situations. In western musical culture, a distinction prevails between the act of composing, and the act of performing music. This distinction has been predominant for several hundred years. These two different positions – composing and performing – are on opposite sides with respect to time. Composing is, at least to some degree, a time-independent activity. In contrast, performing music happens in time; it is impossible to think of a performance as having no relationship to time whatsoever.[3]

These two facets of musical competence also take rather contrasting attitudes toward the skills of their craft. Though thinkers about music who follow Arnold Schönberg's line of aesthetics cherish the idea of practicing the skills of their craft, the skills in question are nevertheless fundamentally different from those of a blacksmith or a shoemaker. Musical composition involves planning of structures and elaboration of material. These activities are essentially non-physical – which adds a dash of metaphor to the notion of "craft". Among various art forms, the architect's work probably comes closest to that of a composer, in the sense that the architect is someone who imagines and draws up plans by relying heavily on the mind's eye. By contrast, a performer working on an instrument, or on his or her own voice, comes much

[3] Let us think of an extreme case: even a performance of John Cage's *4' 30''* does in fact happen in perceptual time.

closer to the traditional concept of mastering a demanding skill in a handicraft kind of way.

In our musical culture of Western art music, a present-day composer may get into a somewhat comparable situation in the case of electro-acoustic music. In such a case, the composer often must develop highly specialized skills in electro-acoustic technologies. But for the most part, the act of composition retains an abstract, non-material quality. In the terms of communication philosopher Marshall McLuhan, performance is a "hot" activity – it fills our consciousness – as compared with composing, which remains more on the "cool" side, inasmuch as it leaves more to our imagination.

It is now time to consider the various roles and mechanisms of musical signs. This calls for an overview of general categories of signs and their role with respect to signification and meaning, and with respect to their objects. We take our point of departure from the semiotic categories of Charles S. Peirce, who defined semiotics as the "formal doctrine of signs" (2.227). One reason to suppose that musical semiotics is not far from the general philosophy of communication is that music constitutes a handy "mini-life" that can aid us in developing more general ideas. Musical messages possess some of the multiplicity and "organicism" present in many aspects of life, but still are nicely limited in scope, and maintain certain properties of formal systems.

To write a computer program that models basic principles of musical semiosis is not a very straightforward thing to do, though it may be tempting to use a computer – the ultimate tool for processing abstract forms – to carry out such a task. There are good reasons for considering semiotics, or semiology,[4] not as a single theory of signs and meaning, but as a conglomeration of various concepts originating in different geographical and scientific worlds, as well as in different disciplines. Various semiotic theories do not always support each other, but at times contradict each other. For this reason, it is impossible to formalize the entire body of musical semiotics for the purpose of computer implementation. One part of the semiotic tradition comes from general linguistics. This may be the easiest part to deal with numerically, whereas philosophical areas of semiotics are by nature resistant to quantification.

Even general linguistics is not easily applicable to musical contexts.

[4] *Semiology* is a term for the French tradition in the study of signs and signification, pioneered by Ferdinand de Saussure. Semiology focuses largely on the same subject matter as semiotics does, but diverges partly in tradition and approach.

Music is not a language, unless one defines language in such broad terms that it is doubtful whether such a definition is of any practical use. In spoken or written language, the notion of meaning differs radically from meaning in music, and in more than one way. For instance, denotative meaning, which relies heavily on the symbolic function, is predominant in the case of linguistic communication. In music, however, the notion of meaning in music is not a simple symbolism. In his book *Linguistics and Semiotics of Music,* Raymond Monelle points out the independence of musical messages from denotation:

> The musical sign is empty, not because of its impotence in referring to real objects, but because meaning is itself fundamentally empty; the sign points beyond itself only to reveal a void" (1992: 20).

Monelle also reflects on the way meaning emerges in music. His view of musical meaning is quite sparse: "Music, in fact, is a tissue of relations only." He seems to believe that musical references to music itself, by means of repetition, are neither valid examples of semiosis, nor of the communication of meaning in general: according to him, it is "logically faulty to describe something as a sign of its own qualities" (ibid.: 209).

The present study, however, takes a different position. Let us consider, for instance, the case of repetition in minimalist music. Each of the repeating patterns in, say, Steve Reich's music may be a replica of many others, but during the repetition process a concept emerges in a listener's mind. That concept is not the repeating pattern itself. More likely, it is an idealized archetype of the pattern. The archetype and its particular instance are not at all the same thing. There is no reason to suppose that a signifier – signified relationship between the two would be out of the question. The rhythm loops – the standard way of building repetitive patterns, for instance, in techno music – could be taken as another example of self-referentiality and of the emergence of a concept through repetition. In those electronic pop music styles relying heavily on samplers, the matter becomes quite evident. Listening to a sample only once produces a quite different cognitive and emotional situation than does listening to the same sample several times in a row. Multiple playings of a sample trigger a different mode of hearing, and the sample, too, is experienced differently. The repetition causes fusion to happen, and a more abstract concept emerges. If 200 patterns are spliced into a continuous succession, they most likely will not be perceived as separate patterns, but as an ongoing process that extends beyond the level of detail. Single elements fuse together, creating a texture that is woven in an ongoing manner.

Martinez, cited earlier, also casts a suspicious eye on Monelle's picture of iconism. According to him, Monelle's description of an icon as having certain properties in common with its object " is a very restrictive view that does not do justice to the amplitude, significance and consequence of the Peircean idea of an icon and iconic sign, and thus leads to mistaken arguments ..." (Martinez 1997: 33). According to Martinez, Monelle greatly oversimplifies Peirce's concept of icon. Because of this oversimplification, Monelle fails to accept the possibility of music representing itself:[5]

> The possibility of sign and object resembling each other to the point of identity is a real possibility which frequently manifests in music, when a composition or performance refers to its object as exactly the same acoustic phenomenon. This is the very basis of the idea of an absolute music. (Martinez 1997: 34)

There are more reasons, explained in the following chapters, to consider repetition as a form of iconism. Paradoxically, the same tissue of relations, which Monelle considers fundamentally empty in the semiotic sense, here becomes the argument for semiosis based on self-reference. In what follows, this "tissue of relations" is referred to as a "network", thus providing a link to the realm of data structures and parallel computing. Before we consider further the properties of this "tissue", or network, it is necessary to introduce some conceptual foundations. Turning to some basic semiotic ideas for assistance, we begin with the nature of signs, especially musical signs, followed by a discussion of musical objects and their possible manifestations.

[5] Martinez's critique of Monelle's view on iconism is considerably more extensive than presented here. Readers interested in learning more about these contrasting approaches is advised to turn to the original sources.

## 3.3 On the Peircean concept of signification

Again we turn to the semiotic theories of the American philosopher and mathematician, Charles Sanders Peirce. His work can not be discussed at any great length within the scope of this study. Peirce's work, a massive collection of reasoning on the nature of signs and signification, is directed to constructing a number of sign-categorizations, both parallel and hierarchical, as well as their analytical functions. Only a small portion of Peirce's semiotic concepts will be exploited here, in aid of framing our own model.

According to Peirce, representation is a process wherein something "stands to somebody for something in some respect or capacity" (Peirce 1955: 99). He introduces four main agents present in the signifying process. The sign, or *representamen*, and its *object* are the most obvious ones. In addition, Peirce distinguishes the *interpretant* as another sign, which is generated by the representamen in the mind of the person receiving it. Representamen, object, and interpretant constitute a genuine triadic relationship that cannot be divided into dyadic ones. Other relationships are joined to this very basic one. One is the relationship between the interpretant and its "ground". About the ground, Peirce says that a representamen stands for an object "not in all respects, but in reference to a sort of idea, which I have sometimes called the ground of the representamen" (Peirce 2.228). Obviously, communication can not take place unless there is a context to relate messages to. Hence, *ground* is the fourth main agent in the Peircean concept of signification. These agents are joined together with relationships, elaborated below, in order to clarify the process of how one agent is generated on the basis of another. Here, we shall call the formalisation of these relationships "procedurizing", since our objective is to formulate general procedures for the analysis and generation of signs.

Peirce's concept of signification strongly emphasizes the immanent mental process in communication. According to him, "thought is the chief if not the only mode of representation" (Peirce 1955: 100). Peirce's reasoning, as concerns context and various factors contributing to information transmission, foreshadow the flurry of communication studies of the mid-20th century. He gives attention to the framework necessary for transmission: as a prerequisite for effective transmission, both sender and receiver, at opposite ends of a communication chain, must share the necessary information-framework for encoding and decoding messages.  A common code is necessary, a fact which Peirce emphasizes:

> ... if there be anything that conveys information and yet has absolutely no relation nor reference to anything with which the person to whom it conveys the information has, when he comprehends that information, the slightest acquaintance, direct or indirect – and a very strange sort of information that would be – the vehicle of that sort of information is not, in this volume, called a Sign.  (Peirce 1955: 100)

## 3.4 Firstness, secondness, thirdness

Peirce's work on signs and signification relies on a three-fold idea of the *mode* of a sign: the modes of firstness, secondness, and thirdness.  This trichotomy, which forms the underlying basis of many divisions and characterizations of signs, is defined by Peirce (8.328) as follows:

> Firstness is the mode of being of that which is such as it is, positively and without reference to anything else.
>
> Secondness is the mode of being of that which is such as it is, with respect to a second but regardless of any third.
>
> Thirdness is the mode of being of that which is such as it is, in bringing a second and third into relation to each other.

Peirce's thought is rather abstract on this matter, but perhaps can be clarified in the following examples (Monelle 1992: 194):

> Firstness is the area of pure possibility.  Before we can perceive a man, it is necessary that such things as men may exist, and that it is possible to perceive them.
>
> Secondness, the most obviously "real" plane, is the area of "happening-to-be"; not only is it possible that a man may exist, but there happens to be a man before me now and I perceive him.  "The real is that which insists upon forcing its way to recognition as something other than the mind's creation".  This is the level of "experience".
>
> Thirdness is the area of purpose, intention, relation, will, understanding, cognition.  When I see that the man is the porter, that he intends to give me a message, that his arrival may interrupt my work or raise my spirits, I enter the domain of Thirdness.

In the context of our study, the categories of firstness, secondness, and thirdness may be situated, interpreted, or implemented in a number of ways. Figure 3.1 diagrams the chain of musical communication, consisting of musical ideas, and its translations and transmissions.  The figure resembles many well known, information-theory inspired diagrams of musical communication; but it is divided in a particular way, with the message and its semiotic process in mind.

In Figure 3.1, musical communication is displayed as a seven-stage process.  Notions of firstness, secondness, and thirdness are interpreted separately on the two ends of communication, i.e., those of sender and receiver. Of course there are more than two participants in this communication.  On the sending side is the role of composer, on one hand, and of performer, on the other.  These roles may intermingle; for instance, in improvisation.  Different roles may also obtain on the receiver's side, depending on his or her mode of listening.

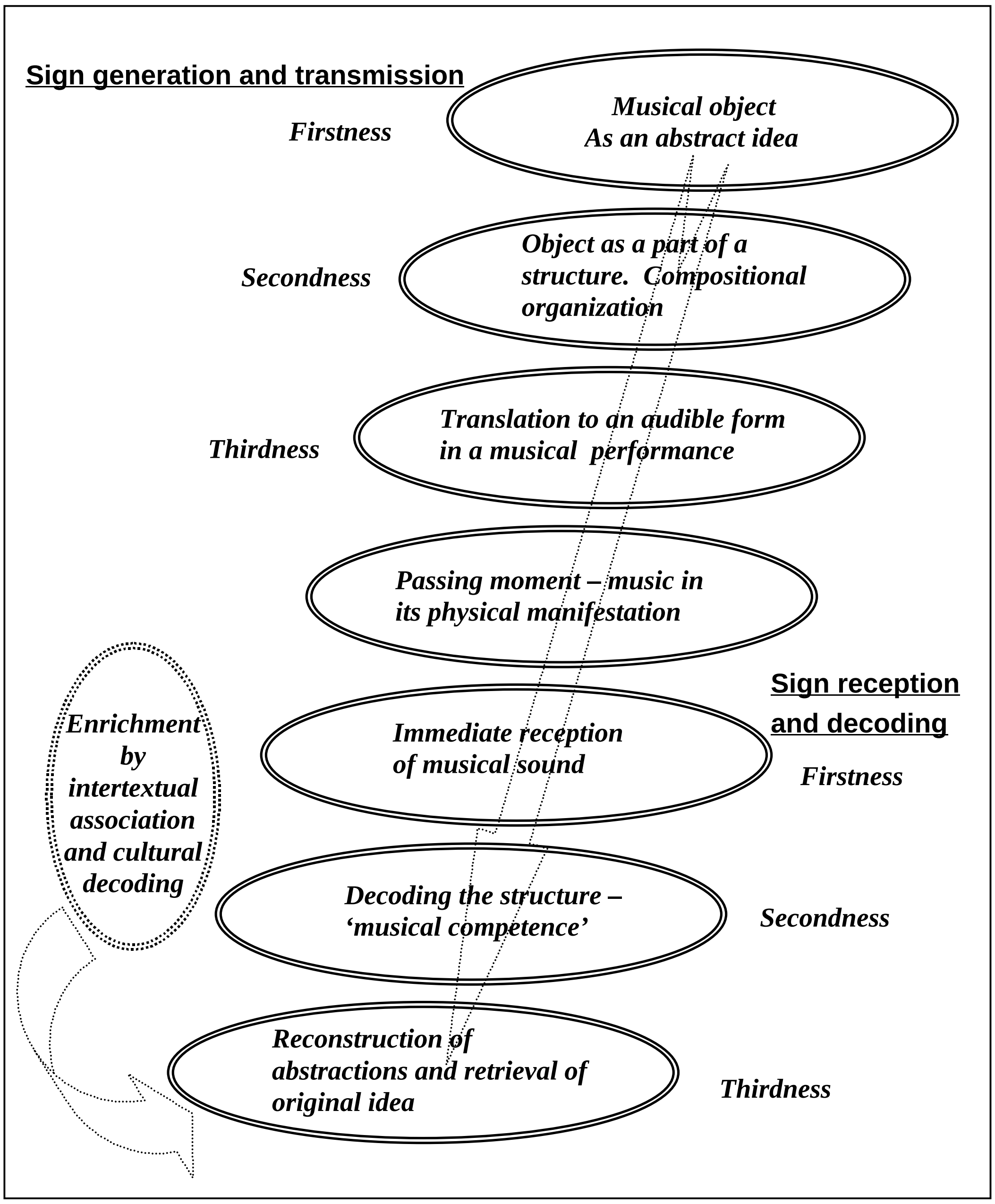

Sign generation and transmission
Firstness
Musical object
As an abstract idea
Secondness
Object as a part of a structure. Compositional organization
Thirdness
Translation to an audible form in a musical performance
Passing moment – music in its physical manifestation
Sign reception and decoding
Firstness
Immediate reception of musical sound
Enrichment by intertextual association and cultural decoding
Secondness
Decoding the structure – 'musical competence'
Thirdness
Reconstruction of abstractions and retrieval of original idea

**Figure 3.1:** Exchange of musical information viewed as a semiotic process.

Figure 3.1 differs from many of those commonly used to depict communication. Such depictions usually show information transmission in the following chain: *composer score interpreter performance listener,* which sometimes includes a feedback line or two (for instance, from listener back to interpreter). The point here is to follow the development of musical ideas and musical objects at various stages of the musical signification process.

The mechanism depicted by the figure has expressions both in the minds of composer and interpreter – as well as listener – and we might choose any one of these viewpoints as a subject of study. Let us first look at the composer's side of the matter.

The transmitting part of the diagram reflects the roles of the composer and the performers. For a composer, firstness manifests itself as an abstract idea of a composition, which has not yet been worked out in any way. It introduces a mere possibility for musical dynamics. If one considers the act of composing as an inversion of the Schenkerian analytic process, then firstness would be some kind of "bud" or embryo, the core idea of a musical work in its primeval form before the act of *Auskomponierung*, or "composing out", has taken place. Secondness, the "real plane" or "happening-to-be" level of musical communication, would reside in the handicraft of composing. This is the stage at which the "tissue of a relations", in Monelle's words, is worked out in detail. This level is where compositional ideas are elaborated, bringing a musical work into existence. Composition of this network of relations gives birth to a major part of the musical meaning of a work. Obviously, the core ideas of the musical message are already present in the domain of firstness, but there are many examples in western art music of a surprisingly complex elaboration of very simple motivic material. To be convinced of this, one need only think of a Brahms symphony. For that reason we are inclined, in this study, to emphasize the generation of musical signs in the domain of secondness, where dynamic relationships between musical objects are established, organized, and elaborated. Thirdness, the area of the composer intentions, may be placed anywhere – from a deep, cognitive understanding of musical *parole* developed in the listener's mind, to any possible extra-musical message that might be motivating the work. It could even be the action of musicians performing the piece. In a very immediate and somewhat extreme sense, it could also simply be the sound itself, the passing now-moment of physical vibrations belonging to musical performance.

For a listener (receiver), firstness, secondness, and thirdness obviously appear in different positions in communication. The now-moment of a physical sound belongs to the mode of firstness. Firstness in music reception also could be extended to include immediate, non-analytic listening. Immediacy is the key concept here, which I would be inclined to call plain *hearing* as contrasted to listening. The immediate nature of this level of reception is manifested in non-analytical reception of sounds as plain sounds. If one were to call it listening, it would be listening in a sort of Cageian sense – hearing with no attempt to "understand" or decipher. Martinez (1997: 31) puts it this way:

> Firstness, in the case of musical listening or appreciation, is when the consciousness of the listener completely blends with the qualities of the music. It is a pure quality of feeling that requires the listener to focus on that which is merely present to the ears.... Hence, to listen to music as firstness requires a naive mental disposition that is commonly found among children, artists and mystics, but that is hardly possible for a mind laden with significations during most of its conscious time.

Listening, in the ordinary sense of the word, already covers the notion of decoding a message, and hence belongs to secondness. Listening, due to the nature of the biological function of hearing, will always be analytical at some level. It involves pattern processing and deals with short-term and long-term memories, since musical patterns are time-dependent and often hierarchical. Any such analysis arises from knowledge of tradition. The objective of developing one's musical listening abilities is to tune or "temper" them, so that they are in resonance with the cultural and aesthetical conventions. Musical education also enables us to decode the technological aspects of a work, in terms of music theory-oriented concepts such as motif, theme, development, counterpoint, and so forth. Such matters also belong to the domain of secondness, which forms abstractions for deciphering the original idea of music. In addition to cultural conventions, mechanisms of cognitive association play an important role in the deciphering process. Later we engage the question of whether or not cultural knowledge can be partly replaced by artificially constructed mechanisms of association, and still arrive at a satisfactory decoding. Computer simulation might help in discovering how much of musical reception – the hearing side – is cultural, and how much depends on cognitive mechanisms. Regardless of how that issue is settled, secondness will beget thirdness, as abstract concepts emerge and reproduce the intention of the work. Thirdness is that level of communication, which a listener may enter by using her cognition as well as her comprehension of technique, aesthetics, and convention. Discussion of musical meaning belongs to this domain. In successful communication, the listener grasps the composer's intentions on this deeper level of musical understanding.

Abstract idea, compositional organization, and performance of the work form the first three links in Figure 3.1. They constitute the transmitter's part of the communication chain. The fourth link, the message in its physical form, is the central one in the figure. It really belongs neither to the transmission or the reception side of the figure. It implements a neutral level, a bridge between the two sides, and is a point where communication lends itself easily to objective, quantitative measurement. Somewhat paradoxically, that is also the point in the chain of communication, which is furthest away from the essence of the musical object – the abstract idea. The first three links in the figure constitute a vector that points at lesser degree of abstraction. Moving away from abstraction, the idea is transformed into physical form; it thus enters into the "real" plane, in the sense that transmission can now take place. The links following the "real" plane on the receiver's side make up a vector leading back to a greater level of abstraction.

A musician giving a performance – a singer, an instrumentalist, or a conductor – would have yet another approach to the modes of musical communication. For a performer, firstness, the area of pure possibility, would most obviously be represented by a score of printed music. In the case of improvised music, firstness could be any plan or sketch made before improvisation, or even a mental state of the musician, such that it would be the major contributor to the performance. Secondness would be represented by learning and rehearsing the piece. Locating the performance of a piece in the trichotomy is somewhat ambiguous. It could be located in secondness, if it is seen as a means of communication; or in thirdness, if it is seen as an end in itself. From the performer's point of view the remaining three steps – immediate reception, decoding of form, and reconstruction of abstractions – would no doubt belong to the mode of thirdness, for they constitute matters of purpose and intention in the performance of music.

## 3.5 A fundamental Peircean triad: Icon, index, and symbol

We next turn to what are perhaps Peirce's most profound and, certainly, his best-known sign classifications: *icon, index* and *symbol*. In his reasoning on icons and hypoicons Peirce says that the "most fundamental [division of signs] is into icons, indices and symbols" (2.275). On the other hand, we must acknowledge that Peirce himself neither regarded this triad as logically the primary one, nor even as the simplest one (2.243 – 2.252). He introduces three categorizations, each dividing signs into three classes. His first categorization deals with the relationship of signs as themselves or with each other: *qualisigns, sinsigns,* and *legisigns*, which we shall return to later. The categories of icon, index, and symbol manifest the relationship of signs to their objects. They form Peirce's second triad. The third triad introduces signs in relation to the interpretation process involved in using them. These classes he calls *rheme, dicent,* and *argument.*

Throughout Peirce's works, icon, index, and symbol receive different nuances of description and definition. Scholars developing applications of Peirce have increased this instability of definition even further. Still, the basic idea remains the same (summarized from 2.247 – 2.249):

> An *icon* denotes its object by common characteristics causing a direct resemblance.
>
> An *index* refers to the object by some characteristic, which it has in common with the object. Yet it is not a resemblance of the object. In essence, the index is a pointer that leads to its object. A common example of an index is smoke as an indicator of fire.
>
> A *symbol* refers to its object by convention. Any such convention is a cultural phenomenon; therefore, a symbol possesses hardly any independence of context.

The classification into icons, indices, and symbols is quite clear as such. But how are these basic categories manifested in music? And in what kind of framework could they be subjected to quantitative treatment and computation? Or perhaps it would make more sense to turn the question around: On what conditions could music-oriented computation be subjected to these categories?

In digital computers, data representations are based on convention, at least in lower-level layers of the virtual machine model. There is a symbolic relationship between the subject matter of computation and its inner representation, i.e., the sign that denotes the subject matter. It naturally follows that the tools and structures, which form the methodological core, either depend on or presuppose the symbolic function. Among such basic edifices of computer science we find, among others, automata theory, formal grammars, stochastic methods of computing and database systems, perhaps even artificial intelligence. Outside this symbolic paradigm of computing not very much would remain besides plain mathematical/arithmetical, "number-crunching" computing.

Iconicity in music representations may take many forms. Considering various music representations, Roads (1985: 406) remarks that they exemplify iconicity to varying. An iconic representation of music may be, for instance, a recording. Music recorded by analog techniques, like reel-to-reel tape recorder, presents an extreme example – the magnetic flow stored to analog tape has its direct correlate in acoustic reality. If the physical and electronic specifications of the storage medium, like tape width, speed, equalization and the like, are fixed, there remains little room for convention between the object, the sound, and the recording that represents it. Physical representations in digital systems are a parallel – the role of convention has been minimized.

Electronic representations of musical and acoustic events came together with technologies that became available after Peirce's time, so no single "Peircean" position regarding them can be the original, correct one. There is, however, a species of visual sign somewhat like them: the photograph. Peirce brings refers to the photograph as a sign several times, sometimes as an index (2.265), sometimes as an icon (2.281). In fact, the case is not one of simple division. For Peirce, an icon is mostly related to firstness; index relates to secondness; and symbol to thirdness. But signs of the first kind often bear some relationship to the second kind as well. Such cases could be called *degenerate secondness*. Concerning photographs, Peirce says this:

> Photographs, especially instantaneous photographs, are very instructive, because we note that they are in certain respects exactly like the objects they represent. But to these a resemblance is due to the photographs having been produced under such said that they were physically forced to correspond point by point to nature. In that aspect, then, they belong to the second class of signs, those by physical connection. (2.281)

Thus, indexical properties are also present in the case of a photograph. Similarly, a music recording, though being a resemblance and therefore iconic, also possesses an indexical quality, which Peirce would have probably called a degenerate one: it points to the performance where it was recorded.

There is at least one major difference between photography and electronic representations of sound: *reversibility*. One cannot possibly imagine how the physical reality serving as the object of a photograph could be retrieved, replicated, or instantiated from the photographic sign. In contrast, with a modern electronic or electronic/optical representation of sound, properly joined with a suitable playback system, acoustical phenomena may be replicated over and over again. Such replication may even reach the level of precision necessary for passing the Turing test – giving a listener the impression that the original sound source is actually present.

Because of the extensive work in categorization, which Peirce conducted, one may feel the temptation to delve deeply into all possible signification situations in all kinds of semiotic categories. It may be wise to abstain from doing so here, though. Otherwise one might take the task of categorization to such a detailed level that it becomes an end unto itself, as a fruitless academic excercise, instead of assisting in the search for fundamental dynamics involved in understanding signs and signification. Rosario Mirigliano (1995: 55) warns against a pretentious interest in taxonomy and detail, which leaves substance totally aside: "almost as though the task of a semiotics of music might exhaust itself in the exercise of seeking the right collocation for musical signs in the network of Peircean trichotomies or some other classification scheme".

Too much love for classification can be counterproductive for understanding semiosis, by making a wider perspective on sign systems and processes unclear. The task of the discipline is not only to bring about ever finely-drawn classificatory schemata, but more importantly, to seek to understand the dynamic nature of signs and their interaction within the framework of such schemata. With this word of warning in our ears, we continue our tour of semiotic categories.

## 3.6 Peirce’s first triad: Qualisign, sinsign and legisign

According to Peirce, a *qualisign* is a quality that is a sign. In music, it could be a particular characteristic of a certain sound or an instrument (the explanation here loosely follows the outline in Martinez 1997: 69). The characteristics of an instrument could refer to any distinct feature, such as amplitude envelope, dynamic spectrum, pitch range, or any other feature of a particular sound. In this case, the quality of the sound also denotes, in an obvious way, the qualities of the sound-producing mechanism. For example, the amplitude envelope may immediately communicate the sound-production mechanism in that particular instrument; the spectrum may also indicate the structure and material of the sound source; the pitch range, which is connected with the physical size, characteristics, and limitations of the instrument.

The role of qualisign can also be relevant in music outside the instrumental paradigm. It could be applied to electro-acoustic music, for instance. A qualisign could be a common factor, a characteristic feature common to a certain family of sounds in *musique concrète*. The spectromorphological categories of Denis Smalley[6] could be considered as qualisigns that extend the concept of instrument-with-a-characteristic-timbre, which would be the case with music made by acoustic instruments (Smalley

[6] *Spectromorphology* is an approach proposed by Denis Smalley for an analytical tool capable of processing any music that is not based on individual notes as building blocks. Smalley’s design was aimed at giving analysts a way to deal with spectrum-based electro-acoustic music. The core idea is of the study of sound objects in terms of their spectral characteristics, shapes, figures, and gestures. The approach also includes a taxonomy of different possible shapes. Spectromorphology can be applied to any kind of music, including classical instrumental music. Besides serving musical analysis, spectro-morphology could aid in compositional activities as well.

1986: 61-93). The notable difference between a timbre and a spectromorphological category, in the semiotical sense, is that the former does not have a relationship to time, whereas the subject matter of the latter essentially concerns motion with respect to time.

A *Sinsign* is an actual instance. In music, it is a performance of the work, a part of it, a period, a theme, a motif, or a gesture. A sinsign may exhibit only one quality; but more commonly, it displays several qualities that come from qualisigns. The behavior or dynamics of sinsigns are governed by *legisigns*, which constitute the framework of a musical culture, subculture, or even a single work. Thus, a typical case of music analysis (even though not the only possible one), expressed in semiotic terms, may be a study of sinsigns, based on the knowledge of qualisigns, with the objective of finding the hidden legisigns in the work.

Given how musical objects are defined in the present study, the fundamental triad of qualisign, sinsign, and legisign seems very useful, though this triad may not be as obviously applicable as that of icons, indices, and symbols. It appears to be related to the mechanics of how the idealized form of a musical object becomes analysed into its different factual instances within a complete, existing work.

## 3.7 A third Peircean triad: Rheme, dicent and argument

The third Peircean triad – *rheme*, *dicent*, and *argument* – reflects the relationship between the sign and its interpretant. The interpretation process is an inevitable part of understanding music. It is therefore reasonable to think that rheme, dicent, and argument are mainly concerned with listening. This accords with Eero Tarasti's (1994) view that rheme, dicent, and argument is the triad most capable of depicting the reception of music. As such, this Peircean triad is the one least relevant to the present study.

In his discussion of rheme, dicent, and argument, (2.250 – 2.252), Peirce describes rheme as a sign of *qualitative possibility*. Along the lines of

Martinez (1997: 70), the musical interpretation of rheme could be a feeling of possible recognition on the preliminary stage, with plenty of room left for a doubt ("Which piece is this? What is the time signature, and what key is this piece in?"). Dicent, a sign of *actual existence*, could in turn be interpreted as factual recognition of a musical work ("This is 'Ornithology', a piece by Charlie Parker"). An argument, "*a sign which is understood to represent its object in its character as sign*" (ibid.), will be created in a knowledgeable, educated-listening situation ("This particular tone is used in order to produce the unusual harmonic situation towards the end of this chorus").

It seems clear that rheme, dicent, and argument involve modes of signification that operate on a very high level of abstraction as far as musical intelligence is concerned. Their extensive exploitation lies beyond the capabilities of the computer implementation presented in this study. As concepts, however, they may prove useful later in explaining the results of the empirical part of our study or future developments of it.

# 4. Adaptation, machine learning, and self-organization

## 4.1 Energy-exchanging and information-exchanging systems

Empirical research, with all its technical details, requires a working knowledge of certain fundamental notions, such as machine learning. To get an idea of how these notions developed, it will be useful first to look at some early thoughts on the foundations of computer architectures, especially ones that involve machine learning. Some of these thoughts come from such famous pioneers of automata theory and cybernetic modeling as John von Neumann. Though published several decades ago, the continuing relevance of such thoughts may come as a surprise to some readers. After all, as seen from the point of view of today's rapidly changing computer technology scene, they seem like ancient history.

In order to study machine learning, one must first lay out some preliminary concepts such as adaptation and complex system. First let us consider the definition of *system* in general. Day to day we deal with various systems – social, biological, knowledge, belief, and many others. A system is separated from its environment by a conceptual boundary that can also be a physical one.  Put more precisely, a system consists of *a set of elements in a relationship with each other*.  Here, a relationship means interaction.  In the literature, most definitions of system usually run along these lines.  For instance, in discussing principles of learning systems, information scientist Mihajlo D. Mesarovic describes a general system the following way, giving it also in the form of formal notation:

‘*A general system is a set of statements*[7] *each consisting of terms and relationships*.  A system is therefore specified in the following fashion

$$\begin{aligned} \Omega_{s1} &= R_1\,[x_1, \ldots, x_n] \\ &\;\;\vdots \\ \Omega_{sm} &= R_m\,[x_1, \ldots, x_n] \end{aligned} \qquad (1)$$

where $\Omega_R = \{R_j\} = R_1, \ldots R_m$ are relations; $\Omega_x = \{x_j\} = x_1, \ldots x_n$ are system terms.’  (Mesarovic 1962: 10).

Systemic studies focus on the interaction of elements that together form a whole, as opposed to the contrasting idea of dividing something into its smallest possible units and studying each one separately.  In addition to its elements, a system has an environment, which serves as a ground that locates the system in a context.  Interaction with the environment basically means a trade or exchange of something in one form or another.  Systems are classified as *open* or *closed*, according to the kind of interaction they have with their environment.  Open systems interact with their environment, whereas closed systems are independent of the environment.  Sometimes it is helpful to define an intermediate stage between closed and open systems – so-called *relatively closed* systems, which have limited interaction with their environments.

Classical physical system theory has been mainly concerned with the exchange of *energy*, as the principal form of interaction between a system and

[7] The following is a footnote from the original text [with the present author’s supposedly necessary correction in brackets]: ‘Term statement is used here very broadly to indicate at the general non-mathematical nature of the relations used in the definition of a system.  If the statements are given interpretations in a logical system [they] are assumed to be true.’

its environment. Lately, this notion has been extended, because in engineering applications software is replacing hardware at a rapidly increasing pace, making physical matter and energy somewhat interchangeable. Consequently, it is often equally suitable to consider exchange of *information* instead of energy as the form of interaction between a system and its environment. With information transfer, as concerns our interests, we will again find ourselves dealing with signals. Energy and signals/information have much in common as concerns system theory. Some information processing architectures (so-called *Boltzmann machines*) have been inspired by thermodynamic concepts, and in some respects, working with them conceptually resembles working with mechanical systems.

Important to our definition of system is that it has a *state*. The number of possible states of a system may be quite limited, or it may be very large. Systems with an infinite number of states are also possible. In energy-exchanging systems, the state is characterized by the level of energy manifested in the system. An energy level can be measured and presented as a numerical quantity, such quantities constituting the *state variables* of the system. The values of state variables contain an unambiguous definition of the system's state. In the case of information-exchanging systems, the state variables follow from the information content of the system. Yet, not all information in the system necessarily has a direct effect on state variables. System control information may exist separately from other flows of information (depending on the definition of system), which may be processed, transmitted, received, or stored by it.

Other distinctions among different types of systems may prove useful. At times one must distinguish between *discrete* and *continuous* systems, often characterized as *digital* and *analog*. In some cases, this distinction can be the same as that between software-based and hardware-based systems, though it is not necessarily equivalent. Tempelaars (1992: 167-180) provides other system classifications that might be useful at this point, such as *linear/non-linear, time-variant/time-invariant,* and *causal/non-causal.*

Tempelaars' classification principles can be summarized as follows. In a linear system, the output signal is in a direct, first-order relationship to the input signal. This means that a constant exists, such that input signal multiplied by constant results in output signal. Generally speaking, a change in a system's input will always cause a proportional change in the output. Many "real-life" engineering applications require keeping the system as linear as possible. For example, linear behavior is often desirable in the construction of electronic appliances, and also in most signal-transfer systems. In practice, however, it is impossible to build an absolutely linear system. Thus, a matter of interest in real-life systems is to define the tolerances that are acceptable and sufficient to enable a close enough approximation for whatever the purpose of the system might be.

The *causality* of a system means that output signal is produced only after input signal has been supplied to the system. As long as there is no input, a causal system cannot produce any output. This comes very close to saying that a system cannot predict the future. *Time-invariance* means that the relation between input and output does not depend on a particular point in time. This is clearly not the case in many electrical devices, such as those, which amplify or transmit sound signals. Various types of time-dependent variances and side-effects may be introduced into real systems. Among these factors might be non-ideal properties of the elements, such as heat build-up in transducers or other components. Thus, time-invariance can not always be taken for granted in sound-signal processing.

## 4.2 Two cases of adaptive systems: Direct adaptation and complex adaptation

The next step on the way to machine learning is the case of the adaptive trait, which might be called direct or *simple adaptation*. Such cases are very common in mechanical control systems; for example, in thermostats. In mechanical engineering this kind of behavior is achieved through the use of *negative feedback*. One of the first tasks which cybernetic research took upon itself, in the 1940s and 50s, was to develop a methodology for working out systems with negative-feedback control schemas in engineering uses. Negative feedback means that, depending on the kind of system in question, either energy, information, signals or ideas, which appear at the output of the system, are fed back to the input, either completely or in part. If built so as to be adaptive, a negative-feedback machine will follow up its own output, and adjust one or more of it is internal parameters in order to achieve a state of equilibrium. Usually the search for stability is the motivation for adaptation. Using feedback for stabilization can be a very simple yet powerful control scheme, for it enables one to build self-regulating systems and self-regulating machines in a very economical way – working on a minimum of energy or information, or on signals of minimum length. There is, however, a drawback to feedback systems. Feedback structures introduce non-linear behavior, and are prone to unstable behaviors if the system conditions are not carefully planned.

The dangers of nonlinear arrangements are most obvious in large-scale mechanical structures; for example, at construction sites or in shipbuilding. Disturbances can cause the system to behave unpredictably; they can, for instance, drive it to self-oscillation, which may prove harmful if the structure is incapable of bearing the stress of such effects. But the downside of non-linearity in systems is not limited to mechanical constructs; all kinds of systems are prone to disturbances of this type. In engineering applications, non-linear tendencies habitually lead to technical catastrophes, both large and small – technical applications are usually meant to be predictable, whereas nonlinear feedback systems might not be. In artistic communication, uncontrollable feedback is usually regarded as a disturbance, due to the fact that it easily causes saturation of the communication channel. The result may be total loss of control over the communication. This, too, is a kind of a catastrophe, comparable to the collapse of a physical edifice.

In considerations of complex systems as opposed to simple ones, it seems pointless to ask where the borderline between simple and complex should be, how complex can simple be; or vice versa – how simple complexity can be and still be called complexity. It is more fruitful to scrutinize the properties and behavior of complex systems. The latter often have a non-linear aspect; but this is probably better taken as an end result caused by the system structure, and not a reason for certain behavior as such. The possibility for nonlinear behavior is brought about by feedback as a primary, system-operating principle, and feedback is tightly connected with complexity of behavior. Kohonen (1989: 250) suggests that feedback is a typical distinguishing factor of the ability to do complex computation. In discussing computing architectures based on parallel, distributed processing, he defends the inherent power and fundamental importance of complex interaction of simple processing elements by feedback properties of many such systems:

> … it has already amply been demonstrated that rather complex pattern recognition, motor control and optimization functions are implementable by circuits which completely satisfy the above definition, especially if certain adaptive effects and transfer delays are included in the interconnections. It is the amount of feedbacks being realized in such networks, as already familiar from the theory of formal automata, which introduces ... complexity [into] computation. (Kohonen 1989: 250)

Feedback properties are introduced by the basic architecture of the computing system, and they are a fundamental cause of complexity. The kind of basic architecture that can make this behavior emerge often includes the *distributed control* and *collective action* of a number of elements.

One can also think of various types of collective phenomena. Some are brought about by a large number of similar elements working in an interconnected and collective fashion, in a global control schema. Others may involve hierarchies among the elements. In large collectives, global control may be loose, and with it the localized grouping and action of subsets of elements. For instance, this seems to be the case in one of the most complex systems we know about: the human brain. In the brain, both localization and global control are present.

## 4.3 On definitions of machine learning

The human brain, as an extremely complex system, possesses one strikingly unique capacity that would be highly desirable for machine implementation in an information system. This is the capability of *learning*. Learning is easily explained in the context of biological organisms, in terms of gaining *experience* and being able to apply it creatively in new situations. To extend the notion of learning beyond living beings is not quite so easy. Do the notions of experience and learning make sense at all in the context of man-made devices and automated data processing?

It sometimes seems that machine learning has been almost as notoriously ill-defined a concept as is artificial intelligence. Like AI, the concept of "machine learning" changes with each new technological advance. Usually, machine learning is defined as a learning system that is capable of changing its course of action according to the input. Essentially, such a change means adaptation on some level, either simple or complex. If simplicity/complexity issues are important here, then it would seem obligatory to clarify the idea of simplicity and complexity, which is not so easily done. It may seem intuitively correct to say that a machine learning system is one capable of changing its action in a complex way, depending on the input signal. But then it would be necessary to define exactly what is meant by "complexity". How simple can an adaptive change become, and still be called a change of action in the machine learning sense? This issue leads us back to a question already adjudged to be a conceptual cul-de-sac: How complex is complexity? In answering this question, one easily arrives at a circular definition, and the whole undertaking of clarifying matters misses its target.

To avoid circular definitions, one might look in at least three directions for further clarification of machine learning. First, one might consider commonly accepted technical solutions that have gained definitive importance. Second, one could consider frequently used principles of operation behind the implementation-level technical specification. This would be a middle-of-the-road choice, which stays in contact with real-world solutions while looking for ways to generalize. Third, one could forget about implementation altogether and attempt to create a more abstract and general formal definition.

With the first alternative, it would be possible to list certain technical solutions for machine learning, and work out a definition taking those as a starting point. In that case, however, one would face the same problem as with the poorly-defined artificial intelligence – with technological advances any definitions based on methodological or technical procedures very quickly become outdated, and new ones have to be worked out constantly. To base a theory on such procedures is almost the same as having no definitions at all. Necessary and sufficient conditions for machine learning will not come from this direction, at least not ones that are expected to have a long life-span.

It is possible to get around the problem somewhat, by generalizing the working principles of specific technical solutions. One possibility is to state that learning systems often have two modes of operation:

(1) the “learn” mode, where the system makes adjustments according to the input;

(2) the “use” mode, in which the adjustable parameters are “frozen”, meaning that the system will not acquire more knowledge, and that only run-time operation using the knowledge already gathered is possible.

Another feature of many learning systems is the ability to retrieve knowledge based on examples, instead of on rigorously specified rules. This manner of operation greatly facilitates the building of a knowledge base.

Some definitions of machine learning involve the notion of a machine improving its results on the basis of its past actions (see, e.g., Kohonen 1989: 250). One must guard against reducing this idea to the simple notion of iteration. The reason for this is as follows: Perhaps the most intuitive way to interpret the improvement of actions on the basis of history would be an iterative kind of problem-solving, such that an approximate solution is gradually made more and more precise, until acceptable precision is attained. Such problem-solving strategies have long been used by mathematicians (e.g., in Newtonian geometry). Those strategies represent the case of a gradual, step-by-step method for arriving at the desired behavior schema, a method often referred to as “iteration” or “iterative algorithm technique”. It seems insufficient, however, to define machine learning as iterative approximation. Learning and adaptation require more than trial-and-error; in any case, iteration does not constitute a necessary and sufficient condition for learning. Iterative approximation seems more intuitively acceptable if learning is considered as a characteristic of (admittedly ill-defined) “intelligent machines”. Still, it is doubtful that a gradual, iterative arrival at a certain body of knowledge would as such make the process more “intelligent” than would a single run of another kind of process. If iteration has anything at all to do with learning as a form of intelligent behavior, the connection must be the supposition that it is possible to build up superior knowledge of a problem with a gradual improvement strategy, rather than in a linear, one-pass process. Many systems commonly called learning machines are often associated with distributed data representations. Prevalent methods of handling distributed data representations involve iterative, self-correcting techniques of acquiring knowledge. Thus, the idea of a self-improving machine, which reaches its computational results gradually in iterative passes, is not out of line as such. But iteration does not make a learning system, even though a learning system can make use of iteration. Essentially, the possibility of feedback in the iterative process is what enables the gradual building of a knowledge base in order to extract more information from the examples. The motivations for gradual improvement of the result, as well as the presence of feedback schemata, are also important for the study of sub-symbolic information processes.

Another way of defining machine learning looks more favorable. It is to look for a more abstract idea, and take a step towards formalism in answering the questions, What is the nature of adaptation? And how might the precise meaning of the term be stated?

To help formalize the adaptive properties of a system, it is useful first to look at a simple case of cybernetic control based on negative feedback, as applied to signs and signals. There is a plethora of work applicable to the processing of signal-based information. The principles for processing such data are the foundations of signal theory (at present, digital signal processing in particular), which might also explain the control of both simple and complex communication systems.

Application of digital signal processing theory to communication has a built-in systemic viewpoint on communication and signification. A powerful aspect of the approach is that the signal-processing network, or transmission system, can be considered as a digital filter. In this way, a highly developed and well-known theory may be applied to communication.

Taking the idea one step further, it ought to be possible to study any data processing system – any computer – as a device that receives, transmits, and transforms signals. A computer program that produces output from the given input takes a signal and produces another one on the basis of the first. However, it is an over-generalization to consider any and all data as a signal. The validity of considering a datum as a signal depends on the semiotic aspects of signals and the types of signs that are formed, that is to say, on the type of semiosis that occurs in the interpretation of the signal.

Hence, in extending the subject matter of signal-processing theory, it becomes obvious that communication makes use of many filters, which appear in many guises. In observing a system, one need only regard its input and output data-sets as signals, and thus formulate a new application of filter technology. If we consider the chain of communication as a filter in the technical sense, we find that the concept of negative-feedback control is not unlike the familiar case of *infinite impulse response* (IIR) filters. The latter kinds of filter make up one of the two principal classes of commonly used, digital-filter architectures, whose applications are reasonably well known and studied. Infinite impulse response filters are based on a feedback loop, which sends part of the signal back to the filter, theoretically ad infinitum.

The concept of filter enables us to evaluate a communication chain in terms of its *transfer function*. A filter system may be characterized by its transfer function, that is to say, the mathematical formulation of the relation between its input and output signals. With the idea of transfer function as a basis, we are now ready to give our own definition of machine learning with more precision than was possible earlier: *A learning-signal processing network is a filter that is capable of adjusting its transfer function depending on the input signal*. This definition can be divided into two parts. First, *a learning system is time-variant*. Second, *its variance in time is connected to information given in the form of input signal.*

The variance of transfer function may or may not depend on the overall variance of system structure. The type of variant transfer-function systems that changes its structure came to be called *self-organizing*. Later, the notion of a self-organizing system acquired a slightly different, more general meaning. These types of systems will be discussed later on.

## 4.4 Supervised and unsupervised learning

There are different kinds of machine learning, according to strategies and technical implementation. As already mentioned, some learning machines have two modes of operation, sometimes referred to as “learn” mode and “use” mode. However, this is not a necessary condition. It is possible to conceive of a learning machine in which adjustments are made during the actual run-time (production use) of the system. Even more fundamentally, two basic kinds of machine learning have been implemented. Probably the most well known kind is *supervised learning*. During the learning phase of supervised-learning machines, both input signal and the corresponding, desired output signal are known. In other words, the supervisor (teacher) has access both to the problem and its solution. A supervised learning system can be used to produce new quantitative information, such as classifications of source material according to a known scheme. The motivation for using this kind of learning machine lies in its capacity to retrieve and store large amounts of data, coupled with an easy way of inputting data. Another motivation is adaptation, even in a limited sense: with data processing architectures for learning, it is possible to build up a knowledge base that can also be applied to situations not strictly taught to the system. On the basis of learned information, a system may develop a limited ability to react to problems that it has not previously encountered. A third motivation, and a major advantage which supervised learning has over a rule-based problem solver/classifier, is that the knowledge in the system may be given in the form of exemplary situations instead of explicit rules. This makes supervised learning system also applicable to situations in which it is impossible to form *a priori* a suitable rule set. In signal-processing terms, these would be situations in which the desired transfer function of a processing network is not known. To avoid *a priori* knowledge may well improve the flexibility of a signal-processing system.

The obvious drawback of supervised learning machines is that they are limited to cases in which it is possible to formulate answers to problems, and in which general, model answers to the problems are already known. If model solutions to a particular case are not available, then supervised learning from examples is not possible.

A second kind of machine learning would consequently be called *unsupervised learning*. Unsupervised learning machines go beyond the problem of an unknown transfer function. They are built for situations and problems in which model solutions are not available beforehand, at least not in sufficient precision for quantification. In other words, for a given input, there is no clear-cut experience of what the output signal of the system should be. Unsupervised learning systems draw the necessary knowledge from structures present in the input signal.

Apart from the fact that the right way to respond to the situation may not be known, it is also possible that no single, correct answer exists. In some situations, good example solutions for finding the sense in a data stream may only be found in the data stream that is being analyzed. A typical application of this kind is a pattern-recognition task, wherein the size and shape of the patterns being recognized can only be found in the signal itself. Such cases occur, among other places, in the context of artistic communication, such that a message (i.e., the work of art), at least to some extent, creates its own world – hence the lack of examples. In cases of highly organized musical signals, absence of knowledge about a suitable transfer-function is probably just a product of the fact that any function which might come into question, and which surpasses the level of banality, must possess a high degree of complexity. Such complexity is manifested in what Hofstadter calls *strange loops* (1979: 10). A strange loop occurs in a hierarchical system, when a move from one level to another results in a return to the same place. Equivalent to strange loops is Hofstadter's idea of *heterarchy* (1979: 134), in that the latter system, too, opposes hierarchy. Heterarchies are common in music of multi-level organization but having no single highest level. Such a system may include indirect recursion – cross-reference among several functions or objects. The heterarchical organizing principle seems to conform with the idea of unsupervised learning. The tangled nature of hierarchy/heterarchy is often present in musical communication. Small surprise, then, that Hofstadter uses music to illustrate many of his ideas.

If the correct example situations are not available, then how is unsupervised learning possible? In other words, how can a machine learn if it has no teacher? In unsupervised learning, the machine's ability to "make sense" of input is dependent on the relationships and structures that the system can gather from the input signal. The architecture of such a system is built so as to react consistently to incoming, structured information. Naturally, this makes the method context-sensitive. Input data may contain larger or smaller amounts of information. If no relationships and structures are present in the input signal, the output of a learning machine will not supply structure that is lacking in the first place: it is a "garbage in, garbage out" situation. Unsupervised learning does not necessarily apply to the analysis of any arbitrary data. On the other hand, the advantage here is that theoretically it is possible to isolate and process even such structures, which are not known to the user of the system. Perhaps the cognitive validity of such analysis can then be questioned. But the proof of a machine is in using – chances are that the unsupervised learning machine is able to extract new interesting information from the input signal.

| **Type of system** | **Conditions for use** |
|---|---|
| Explicitly defined rule system | The desired transfer function is known and can be formulated |
| Supervised learning | The desired transfer function is not known, but examples reflecting it are available |
| Unsupervised learning | The transfer function is not known, nor can it be retrieved beforehand from examples |

**Table 4.1**: The relationship between data-acquisition strategy and the definition of transfer function of a processing system.

To summarize, the strategy for building a knowledge base capable of dealing with signals in a meaningful manner depends on the way, how the transfer function can be defined. There are three basic alternatives for such definitions, which are described in Table 4.1.

## 4.8 Another look at sub-symbolism

In the second chapter (2.4.5 – 2.4.8) references were made to *sub-symbolic* data representations. The basic statement there was simply that sub-symbolic representation is a data description of a level lower than that of symbols. The notion of sub-symbolism now receives closer attention.

Sub-symbols in musical contexts have been explained by in term of cognition. For instance, Leman and Schneider (1997: 18) argue that a sub-symbolic approach to musical representation starts from sounds, and from representations based on auditory images and neural networks. This explanation is probably right as such, but as a basis for, say, a software engineering project, one would need a more explicit definition.

The oft-cited paper by Smolensky seeks to define sub-symbolism in a more precise way, in order to take the study of data representations further. At first, the matter is outlined rather vaguely:

> The name “sub-symbolic paradigm” is intended to suggest cognitive descriptions built up over entities that correspond to *constituents* of the symbols used in the symbolic paradigm; these fine grained constituents could be called *sub-symbols* and they are the activities of the individual processing units in connectionist networks. (Smolensky 1988: 3)

From a strictly “hard fact” point of view, the first problem with this outline of sub-symbolism is that it is made through negation. In the above quote, Smolensky essentially defines the sub-symbolic paradigm by contrasting it to the symbolic paradigm. Another problem with the above definition is that it seems, at least to some extent, to connect the whole notion of sub-symbolism to a certain branch of technical solutions, if the statement about “individual processing units in connectionist networks” is taken as an engineering solution. An apology for Smolensky’s definition might be that the model for these particular solutions is immanent, or mind-oriented. On the other hand, connectionism also clearly refers here to the engineering approach to sub-symbolism. This in turn leads to the idea that sub-symbolic data representation is here being defined as one that is used in technical appliances, namely, connectionist computer networks.

We might agree with Smolensky, however, that sub-symbols are the constituents of concepts that develop into a symbolic level in our minds. As an aid to studying the nature of sub-symbols, it would be useful to explore the origins of the symbolic function. This might involve a closer look at Peirce’s semiotic conception of the symbol. Before doing so, we shall return to Smolensky, who further explicates his notion of sub-symbolism:

> Subsymbols are not operated upon by symbol manipulation: they participate in numerical – not symbolic – computation. Operations in the symbolic paradigms that consist of a single discrete operation (e.g., a memory fetch) are often achieved in the subsymbolic paradigm as the result of a large number of much finer-grained (numerical) operations. (Smolensky 1988: 3)

Now we have arrived at a more fundamental and more generally applicable idea. *Computation with sub-symbols is essentially computation with numbers*, as contrasted to computation with symbols. This distinction is valuable for a quantitative approach, and well serves the point of view of this study. To take sub-symbolic operations as numerical operations combines well with an idea introduced earlier: to conceive of musical signs, and other data that originate in music, as signals.

## 4.6 Self-organization: Various definitions in historical perspective

Notions of learning and adaptation are closely related to the concept of *self-organization*. As has been the case with machine learning and artificial intelligence, many attempts have been made to outline and define self-organization. The idea of self-organization emerged in the 1950s and 60s, during the first wave of neurophysiology-driven research into the foundations of the recently born computer science. The great vision of self-organization research was inspired by the build-up of knowledge and skills observed in living organisms. An essential result of such observations is that, unlike the memory of a digital computer, the mind is not a passive storage space for data and experience. Instead, it plays an active role in "making sense" of the world, even though this activity occurs in the "back" of our minds, and we may not be aware of it most of the time. "Making sense" here essentially means creating abstractions and organizing raw data into concepts. In the biological context, self-organization is that part of the learning process which takes place in our minds after the actual learning situation. It takes place during the time necessary to "digest" new information. Arranging data in a particular order is part of self-organization, and building abstract concepts from the new order is an inevitable second step.

Technical implementations using self-organizing methods are normally employed in data-driven classification and recognition functions. In such systems the course of action depends on the data input. The strategies for classification and recognition of features in the input data-stream build upon the contents of the data-stream itself. This type of working principle was considered highly desirable as early as in the first stages of the development of learning machines. Therefore, as far back as the 1960s a good bit of thought went toward finding the operating principles and technical solutions for implementing this property in a computer system.

With the consequent development of machine learning and pattern recognition, technical means of implementing self-organization have continued to be an important research problem in efforts to construct learning machines. Many attempts have been made to define these means on a general level. In their extensive monograph on cybernetics, Klir and Valach (1967: 413) proposed a development of new properties in an abstract machine. In essence, they defined self-organization in terms of the enhancement of performance. They present a self-organizing system as one that is in a process of continual improvement – one might say, a dialectical process – of relations with its environment.[8] According to the authors, after a self-organizing system has been built, it can "later improve itself, and … acquire properties which were not even thought of it when it was created" (Klir and Valach 1967: 414).

Going back to Mihajlo D. Mesarovic's definition of a system as a collection of terms and relationships (Chap. 4.1), we are now ready to study his idea of self-organization. As a reminder, it may be helpful to restate here the equation that was given earlier:

"

$$\begin{array}{lcl} \Omega_{s1} & = & R_1 [x_1, \ldots, x_n] \\ . & & . \\ . & & . \\ . & & . \\ \Omega_{sm} & = & R_m [x_1, \ldots, x_n] \end{array} \qquad (1)$$

[8] Whether or not there are political overtones to be found in their view of self-organization is not important for our concerns here. It might, however, be interesting to search for the roots of the hypothesis among socialist ideas of science, life, and culture in Eastern Europe during that particular period of time. With all the connected belief of positive progressive development of systems, the idea of self-organization could have been thought to apply to societies as well as to technological systems.

where $\Omega_R = \{R_j\} = R_1, .. R_m$ are relations;
$\Omega_x = \{x_j\} = x_1, .. x_n$ are system terms." (Mesarovic 1962:10).

Relying on this equation, Mesarovic defines self-organization as follows:

> Definition 6. Given: (a) A system as defined in (1). (b) A set (finite or infinite) [of ] relations $\Omega_R = \{R_j\}$ defined on the set of the systems terms $\Omega_x = \{x_j\}$.
> A self-organizing system is a system which starts from their initial state as a given in equation (1) and changes its structure by using a relation from the set $\Omega_R$. (Mesarovic 1962:11).

Mesarovic lays down two basic characteristics of self-organizing systems. (1) They are discrete systems. At the time, this was a rather evident conclusion. It was generally understood that the kind of changes that would take place in self-organizing systems, would be structural in such a concrete way, such that it would be impossible to imagine the changes happening in a continuous space. The second characteristic given by Mesarovic was a little surprising: (2) Whether or not the system appears as self-organizing depends on our knowledge of the system variables. In essence, the presence of self-organization is a relative matter. Both of these characteristics are proven in his text.

With Mesarovic's first characteristic, however, the proof did not last long. Later, systems were developed that did not have to fulfill this requirement, but still exhibited behavior considered to be self-organizing. This is not to say that his proof was wrongly thought out. Rather, it reflects the way the notion of self-organization has changed. Current self-organizing maps (SOMs), discussed at length later on, order input signals topologically by strengthening or weakening connections between certain parts of the network. In this way a major structural change can take place in a continuous manner instead of completely breaking up an old structure and building a new one. The older definition of self-organization focused on things that happen to the processing system. Newer notions tend to focus on what happens to the arriving signal, and on what happens to individual components of the system.

Mesarovic goes on to distinguish two types of self-organizing system.

The first one he calls *causal* self-organization. His causal system corresponds approximately to a supervised learning machine. According to him, a more advanced form of self-organization, the *teleological* approach, aims at building machines that constantly enhance themselves. Mesarovic seems to refer to a kind of artificial growth process, paving the way to artificial life. He was probably aware of John von Neumann's interests along the same lines, and wanted to elaborate or augment the latter's work. For that reason, von Neumann's thinking on self-organization is the next item on our tour of learning machines.

## 4.7 Self-organization – the neural model and the evolutionary model: John von Neumann and the origins of cellular automata

In his *theory of self-reproducing automata,* John von Neumann lays groundwork for adaptation and self-organization. Taking biological cell structure as a source of inspiration, as one possibility for building a self-reproducing automaton he outlines a structure consisting of a lattice of discrete elements. He describes the discrete structure of the lattice as *crystalline* (von Neumann 1966: 103).

In a commentary on von Neumann's text, his editor Arthur W. Burks[9] summarizes the work around the question of how self-organization could take place in an artificial environment of automata. The machine outlined by von

[9] Von Neumann's editor, Arthur W. Burks, was interested in the two different disciplines also engaged by the present study: Peircean semiotics and the theory of automata and information-processing systems. Among other occupations, Burks worked as one of the co-editors of the *Collected Papers of Charles Sanders Peirce*. Perhaps his interests were not as diverse as one might think, however. At this early stage of computer science, matters concerning system structure were not a subject for "hard" science and engineering only. They formed a common subject of interest for philosophy as well.

Neumann has the topology of a cellular structure: a network. It starts off as an homogenous and isotropic lattice. Isotropy means here that, at first, the cellular structure of the machine is similarly structured in all directions of space. Specialization, or structure, then emerges in the self-organizing process:

> Consequently if different cells of the region of the cellular structure are in different states, one part of the region may act in one way and send information in one direction, while another part of the region acts in a different way and sends information in a different direction. (von Neumann 1966: 106)

Specialization of different parts of the lattice is a functional trait of biological organisms, and von Neumann considered it to be an important one. Adapting to particular situations meant striving to acquire capabilities that organisms generally have, such as flexibility and tolerance for error. Specialization as an operating strategy might be achieved by introducing *competition* between different parts of the system. In research on biological neural networks, the term *inhibition*, or lateral inhibition, is used to describe such competitive circumstances in some architectures. The term refers to what happens when an active component of the system suppresses other components that are less active. Competition between several subsystems would cause an automaton to have a highly nonlinear transfer function, if properties requiring such were called for. In the context of von Neumann's crystalline lattice, or "self-organizing layer" as it has been called on other occasions, this could mean a build-up of different hierarchical levels. Local and global control schemas could coexist and interact. Thus, behavior of a high degree of complexity could be made possible. Von Neumann attributes this behavior to the autonomy of subsystems:

> The ability of a natural organism to survive in spite of a high incidence of error (which our artificial automata are incapable of) probably requires a very high flexibility and ability of the automaton to watch itself and reorganize itself. And this probably requires a very considerable autonomy of parts. There is a high of autonomy of parts in the human nervous system. This autonomy of parts of a system has an effect, which is observable in the human nervous system but not in artificial automata. When parts are

> autonomous and able to reorganize themselves, when there are several organs each capable of taking control in an emergency, an antagonistic relation can develop between the parts so that they are no longer friendly and cooperative. It is quite likely that all these phenomena are connected. (von Neumann 1966: 73)

In the work of von Neumann and Burks where, this is where the concept of self-organization is used. It does not receive much attention elsewhere, at least not as much as their primary theme – self-reproduction. Still, the thoughts presented throughout their work are highly pertinent to machine learning and self-organization. There are two basic ways to interpret the ideas of von Neumann and Burks. One could think of an interpretation modeled after neural or after cellular/evolutionary paradigms of self-organization. These two interpretations have a great deal in common. No matter which one is accepted, the model connects closely with biology, and even closer with collective phenomena, which are the kind primarily in question here. At least two major interpretations of self-organization are possible: one is neural; the other, cellular. The former has led to research on artificial neural networks, and the latter to models of artificial life and cellular automata. Both paradigms grew out of the same ground.

Von Neumann’s text has many points of interest, all of which are not possible to deal with here. One of the more absorbing ones is his distinction between so-called *crystalline* versus *Euclidean* types of automata. At the level of implementation, this distinction corresponds roughly to the digital and analog worlds of electronic devices; at the level of quantitative description, it corresponds to the difference between continuous and discrete mathematical tools. In some respects, von Neumann seems to consider continuous, analog computing architecture to be the superior one. He compares the two in the following way:

> (X) The general possibilities are about the same in the two cases.
>
> (Y) The continuous case is mathematically much more difficult than the crystalline case.
>
> (Z) If and when the appropriate analytical methods to deal with the continuous case are developed, this case will be more

satisfactory and more broadly and relevantly applicable than [is] the crystalline case. (von Neumann 1966: 106)

Just as in this early work on automata theory, some later neural network engineers have, at least in principle, advanced analog technology as an alternative to digital devices. Indeed, some of the earliest attempts to construct artificial neural networks utilized analog devices and elements, such as potentiometers conceived as adaptive "organs". In some cases the potentiometers were driven by small electrical motors, which would make small incremental adjustments according to the learning schema built into the device's architecture (Alexander and Morton 1990: 57-58). Reports on other, more advanced endeavors using analog hardware have been recorded. In most cases, however, practical issues such as low cost, availability and reliability have called for digital technology, which has had the edge over its analog counterpart in data-processing applications ever since LSI technology became available.

John von Neumann's theory of self-reproducing automata is a good example of early computer scientists' tendency to model artificial systems after natural ones. In von Neumann's time it was generally considered obvious that biological organisms might well be called "automata", just as well as man-made ones were; and, at least on some level, both could be fitted to a common theory. In this spirit, it was quite feasible to speak about artificial and natural automata as parallel cases of basically similar phenomena. Both types of automata could be subjected to comparative study (von Neumann 1966: 21-22). From this background it is easy to see why von Neumann suggested a combination of analog and digital information-processing systems as an architecture for a computer of the future. Indeed, he went on to lay the groundwork for a self-reproducing machine. Arthur W. Burks continued such work, which was left unfinished at the time of von Neumann's death. Burks examines the case of von Neumann's cellular structure, and finishes the work by using logical deduction to show that "self-reproduction is a special case of construction, and construction and computation are similar activities" ... (von Neumann 1966: 296). On the way to this result, many basic ideas of modern neural computing, such as collective parallel computation, competitive learning, lateral inhibition, interaction of digital and analog domains, as well as interaction between global and local schemata, came forth, at least in their preliminary forms.

## 4.8 Architecture of a modern self-organizing system: The two-dimensional self-organizing map

Among modern self-organizing algorithms put to musical use is Robert Gjerdingen's ART system, mentioned in Chapter 2, based on Grossberg's *adaptive resonance theory*. The *self-organizing feature map* as conceived by Kohonen (1989: 119-157) serves as the basis for the experimental part of this study. Thus it is necessary to explain it in detail here. It also serves the purpose of this study as an example of a contemporary self-organizing system. To date it has gathered a following, as a widely known neural network-related approach with some appealing features. It is strictly a technical solution to an engineering problem, but can be also regarded as a model of specialized (localized, self-organized) functions found in the brain – providing one understands that it is just a model, a simplification that illustrates general properties. Here, "specialization" does not mean that the nodes of the network must be physically or formally different from each other, as was supposed in many early theories on self-organization, such as the one by Mesarovic (explained in section 4.6). Rather, it means that the nodes are more or less firmly configured for certain tasks (Kohonen 1989: 119). In a self-organizing system those nodes, which are configured for the same or similar tasks, form local clusters of response. In his study of present-day self-organizing machines, Jacob Marinus Jan Murre recognizes similar operating principles behind a considerable number of quite different systems:

> We may, thus, conclude that the same basic principles reoccur in quite different models, and that they appear to be central to the concept of self-organization. Extending a local selection principle, such as a winner-take-all competition with a sub-local ordering principle, results in a topological ordering of representations. (Murre 1992: 59)

Superficially this localized response may seem to contradict the idea of a parallel distributed representation of knowledge. If a representation is distributed, how can it be localized at the same time? Localization in self-organizing feature maps (SOMs) is nevertheless rather different from, say, the perfectly local one-byte-per-one-memory-slot scheme of conventional (so-called von Neumann-style) digital data-processing. Even though localization and specialization may be found among the nodes, each local focus of activation consists of a number of nodes acting in parallel. The operation of a SOM is built on the interaction of nodes within a defined neighborhood, and on edge-effects of neighborhood clusters. This is what Murre calls "sub-local ordering".

Our treatment of SOMs is restricted to two-dimensional maps, in the spirit of the original examples of Teuvo Kohonen, author of the architecture (Kohonen 1989: 122). There is no reason, however, why one should be limited only to two-dimensional cases. British computer scientist Igor Alexander has given an example of the method being simplified to one dimension (Alexander and Morton 1990: 148-152). One could conceive of three or more dimensions, too. It must be noted that the implementation of SOM explained here is in some respects simpler than other ones found in the literature. Nevertheless, the distinguishing features presented here should suffice to qualify it as a SOM system following Kohonen's ideas.

The SOM used here is a layer (two-dimensional array) of elements, or nodes, connected laterally. Their connectedness gives rise to information feedback. The strength of the feedback is controlled according to a determined shape or function. The degree of connectedness among the nodes – the number of nodes in direct interaction with any single center node – of the layer depends on the implementation. In the present case, each node is logically connected to eight surrounding nodes, and the texture of the layer is essentially a square lattice with both rectangular and diagonal connections. The surrounding nodes form a *neighborhood* around the center node, which is the key element in the making of a sub-local ordering. The immediate neighborhood-circle around a node can in turn be surrounded by a larger circle, and this second one with a third, larger circle. The distance from the center determines the magnitude and evidence of lateral feedback gain, as determined by the feedback function. The shape and magnitude of this function largely determine the behavior of the system. A number of shapes can come up for consideration. Kohonen (2001:

178-179) presents an optimal function-shape for lateral feedback, excitatory from the middle, with an inhibitory circle around, and carrying a diminishing inhibitory effect to the surrounding areas according to the growing distance from the center. Because of its shape, it has sometimes been called the "Mexican hat" function. In our implementation, the "Mexican hat" function has been replaced with a somewhat primitive one-, two-, or three-stage function for the sake of simplicity, and for the computational shortcut it provides. Figure 4.1 represents these basic shapes.

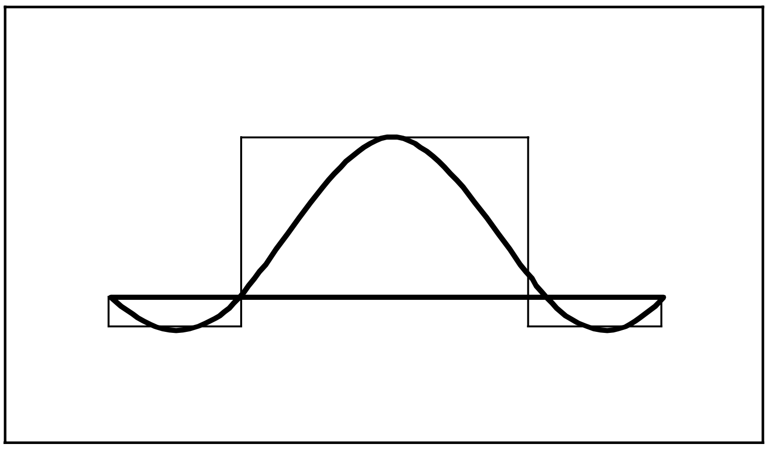


**Figure 4.1:** Approximate shapes of a smoothly curving "Mexican-hat" function and its equivalent step (square) function (see Kohonen 2001: 178-179).

The map, or *Kohonen layer* as it is sometimes called, can automatically adjust to develop specific responses to various patterns in given data. This is achieved in the following way. Data is fed into the map as packages with length N called *feature vectors* $X = \{ x_1, x_2, ... x_n\}$. Each of the nodes has an input data vector $A = \{a_1, a_2, ... a_n\}$, where the data package carried by the current-feature vector is stored.

The nodes also have a representation of their internal state in the form of a vector $W = \{ w_1, w_2, ... w_n \}$. Elements of the vector W are often called the *weights* of the node. This is in accordance with the concept of weights as the primary adjusting elements in various other schemata of artificial neural networks, where tuning the system with weight adjustments makes adaptation and learning possible. Vectors X, A, and W obviously all have the same length

N, which corresponds to the dimension of vector space. As data packets in the form of feature vectors X introduced to the layer inputs A, node receives an identical input, which is processed in a parallel way in all nodes of the layer.

At the initial stage each weight $w_1$, $w_2$, ... $w_n$ in every weight vector W is set to random values. Subsequently the whole corpus of data is chopped into feature vectors X, which then are introduced to the layer. At the learning stage, adjustments are made to the weights as follows: when the nodes receive a feature vector, they develop a reaction to it in the form of a response value Y. The strength of the reaction depends on the correlation between the weight vector and the incoming data vector. The stronger the correlation, the higher the response value (Y) generated by the node. Let $Y_{max}$ be the chosen maximum possible value for correlation. Different formulas could be proposed for calculating the correlation value, for instance, as in (2):

$$Y = \sum_{n=1}^{N} Y_{max} / ((1 + |a_n - w_n|) \times N) \qquad (2)$$

Formula (2) is arbitrary in the way that, it does not describe a mathematical law, but is simply tailored to produce $Y_{max}$ with identical vectors. It must be noted that (2) contains a nonlinear term of the form 1/x.

In the implementation of the experiment described in chapter 7, a simple algorithm is used instead of (2) as to both produce maximum response with identical vectors, yet maintain system linearity. It works as follows:

$$Y = \sum_{n=1}^{N} k\,|a_n - w_n| \qquad (3)$$

where multiplier k is algorithmically chosen so that the highest value found in the data will produce maximum response $Y_{max}$.

At the first stage of adjustments, the weights are set randomly, so when the feature vectors arrive at the nodes, there will be one “winning” node with the strongest response, i.e., the best match of weights. An adjustment is then made to the weights, which strengthens this correlation even more. The winning node will be taken as a center for the second stage of adjustments:

lateral feedback. The weight vectors in the neighborhood of the center node are either drawn closer or pushed further away from the weight pattern of the center node, according to the value of the lateral-feedback function at the respective location on the map.

Thus configured, the SOM will build an order to any given feature-vectors during the teaching process. Consequently, similar (according to a specified metric) features have a tendency to focus the strongest response toward adjacent nodes. Either a static or a dynamic lateral-feedback function controls the degree of parallelism in the network. Properties and structures present in the input data cause a respective state-pattern to develop in SOM. The result is that sub-local ordering is achieved together with a distributed collective representation of knowledge. As Ritter (1991: 379) remarks, “networks with random topology may provide a natural structure for creating topology preserving maps of semantic items”.

The process is called “self-organizing” because no outside supervision is used in forming representations. The map is also called “topology-preserving”, because of the tendency of similar features to trigger activation in adjacent nodes. Thus *clusters of activation* (lateral feedback being responsible for the clustering phenomenon) are formed, subject to correspondence between the topology of the input data and that of the response patterns of the map. The map accommodates incoming data by making small, stepwise adjustments iteratively in a loop. Adjustments are made slowly so as to allow time for the response foci to settle into place and stabilize. The step of adjustment is decreased during the process, so that the later iterations change the weights less and less. This causes the process to stabilize slowly into a certain configuration. A large number of teaching cycles is typically needed to produce a self-organized order. Without the necessary amount of teaching to stabilize the map, the results may include a significant random element. Because of the highly iterative nature of this process, SOM systems have considerable need of computing power.

An additional remark is due, of a semiotic nature, concerning self-organization as it is outlined here. A great distinguishing factor between symbolic and sub-symbolic computation seems to be that, in the sub-symbolic domain (as opposed to the symbolic one) there is an organic relationship – in fact a strong dependency – between a representation and its context. One cannot exist without the other. A context for an individual sign is the signal, of which the sign is a fragment. The signal serves as a *ground* for a sign.

# 5. Musical information as a fabric of sinsigns and legisigns

## 5.1 Symbol, icon, and sub-symbol revisited

We can now revisit the concepts of symbolism, sub-symbolism, and iconicity. The relationship between sub-symbolic representation and the iconic sign is a special one. Both are intimately connected to symbols. According to Charles Sanders Peirce, symbols result from iconic processes: "In the earliest form of speech, there probably was a large element of mimicry. But in all languages known, conventional auditory signs have replaced such representations. These, however, are such that they can only be explained by icons" (Peirce 2.280).

Consequently we can ask, What properties enable icons to explain symbols? Icons are rich in meaning that goes far beyond mere recognition of their origin and object, as Peirce has remarked: "For a great distinguishing property of the icon is that by the direct observation of it other truths concerning its object can be discovered than those which suffice to determine its construction" (Peirce 2.279).

An icon can convey multiple levels of meaning in addition to the main, denotative one because of its direct resemblance to its object. The icon preserves the relationships and proportions of its object, which carry meaning and may allow for various, parallel interpretations.

When an icon appears in various time locations in a message, the multiple roles and facets of its meaning are made clear to the receiver of the message on the basis of the context in which the icon appears. The meaning of an individual sign is explicated and precisely focused by its usage in the signal. In this process, concepts take shape and "find their place" in the semantic space that is created by the vocabulary manifested in the message. Not only do existing concepts come into focus and take shape, but new concepts may also *emerge* from iconic semiosis. As an example, consider electro-acoustic music that is produced in a studio. A composer may have a detailed structural plan of a composition before going into the studio, based on ideas worked out earlier. Before hands-on work on the sound material begins, the plan remains abstract to some extent – meaning that it relies largely on symbolic representation. But when a composer begins working with actual sound material, he or she enters the non-symbolic domain of semiosis. The work may take new and unforeseen directions, because the interaction of creative mind and sound material evokes associations. Resemblance and similitude play a major role in bringing out these associations. They are the basic ingredients of iconic semiosis, and they guide artistic creation.

In the iconic semiosis of artistic work, the *ground* of the concepts or objects being reflected upon is also very important for the development of associations. In our example, the ground would be the composition – the sounding environment, which lends itself to evaluation and to the shaping of each new motif, pattern, or event. Each new element is contrasted against the whole of which it is a part.

Other examples of sub-symbolic semiotic processes that carry and give birth to symbolic notions can be found throughout our culture. One need only think of the development of written alphabets, from the most primitive forms of mimicry to the symbolic, yet cogent, marks of the modern alphabet.

One might argue that the primary nature of association is not iconic, but indexical. This is so because, in an associative process, one concept or notion triggers another, which points back to the previous one, thus forming a chain of pointers. If one uses Peirce's ideas, however, one must accept the fact that, on his view, iconic associations are not indices. He insists that an index is not a resemblance of its object. Indices may arise from iconic situations, in which case they nonetheless depend on a primary layer of iconic signs.

A dependency similar to that between symbols and icons is formed in sub-symbolic data processing, of which numerous examples exist. For instance, Leman and Carreras (1997: 153-155) give an example of a sub-symbolic process in which commonly used notions of music theory can emerge in a completely data-driven way, on the basis of examples taken from cadenzas of tonal music. In their experiment, sub-symbolic data produce a higher-level concept; an essentially new idea emerges, namely, a general construct representing the circle of fifths, one of the very fundamental theoretical structures in the Western musical system. In general, a new abstraction is born in sub-symbolic self-organization, because a new ordering of information emerges as structures take shape in processes comparable to the one investigated by Leman and Carreras. Such processes can therefore be used for data classification.

Organization of a connectionist network, be it self-organizing or supervised, provides data reduction. This is an abstraction process that transforms an iconic representation such that it increasingly gains a symbolic quality. Digitized sound is already a symbolic translation of an iconic representation, in the sense that digits are symbols of numbers, and numbers are symbols of measurements of sound-pressure levels (a fair amount of iconism is involved in PCM-coded sound signals, too). One must realize, though, that numbers belong to a rather peculiar category of symbols. They are sinsigns that signify immaterial entities, namely quantities, and that have an intimate relationship with an extremely formal and elaborate system of legisigns, which includes arithmetics, algebra, function analysis, calculus, and many other kinds of mathematics. Therefore, formal treatment of numbers is rather different from that of generic symbols. This difference is manifested, for instance, in the difference between symbolic computation and algorithmic computation. Thus, digitizing as such does not raise the level of abstraction in a message. If the digital information were compressed, it would be more justifiable to talk about an abstraction process, since data-compression accomplishes an actual reduction in size. In the abstract sense of the word, reduction must be understood in terms of the content of the message. It makes little sense to claim that sound digitized with a lower resolution or sampling rate is more abstract than sound recorded with better quality. In such a case, information is lost, while no new and pertinent, higher-level information is created. Part of the meaning of the message is potentially being destroyed. A central concept here is *reversibility*: how easily and faithfully can the original sound phenomenon be retrieved from its representation?

Not all analysis is reversible, in the sense that the original object of analysis may be recreated from its representation. For instance, in Schenkerian analysis, which produces reductions of massive compression ratios, the reversal process – i.e., synthesis of the original music from analytical results – is neither a simple nor obvious thing to accomplish. But even Schenkerian music analysis makes use of the concept of "composing out" (*Auskomponierung*), which suggests that the process might be reversible, at least theoretically, provided that information is available about how the reduction was accomplished at the first place. Similarly, even though Roman-numeral harmonic analysis is not a completely reversible representation, in the sense that the melody may be retrieved from it, there are cases in which that would be possible. Let us consider, as another example, the harmonic structure in a well-known jazz standard, Morgan Lewis's "How High the Moon". Because in this case the harmonic structure of the piece is clearly recognizable, any musician with a little education in jazz would be able to apply his or her knowledge, and play the melody upon just seeing the chords. Both of these examples suggest that it should be possible to reverse an analytical process, at least in principle, even though extra information might be required in order to do so.

Musical scores have iconic properties, but they are also partly symbolic. Graphic scores and tablatures obviously depend more on iconic semiosis than does common-practice musical notation, which relies mostly on symbolic signification. In fact, common-practice musical notation is symbolic enough to allow its iconic properties to be ignored in fully functional formalisms. Numerous computer-based notation and sequencing systems prove this last statement to be correct. Computer sequencers, with their plainly symbolic representations, fill many needs of present-day commercial music by virtue of their descriptive power. Use of this machinery comes with cultural consequences: symbolic processes in music technology have had an especially pronounced effect on the development of current popular music. The musicians' gestures and physical presence are distanced from the act of music-making, for they no longer need to exist in real time with the performance or recording – sometimes they do not need to exist at all. Music may be cut, copied, and pasted as tracks, blocks, and events, and may be step-edited at will, in a manner quite remote from the way musicians actually perform. Such a development affects reception, as well as general aesthetic principles. This distancing effect is most pronounced in popular music, which is culturally significant, because today it is the most influential type of music for the majority of people. Cases that may be only slightly less apparent can be found in concert music as well; for example, the characteristic sound of a certain studio or group of performers. Symbol-manipulating technology thus in turn manipulates musical communication and semiosis on a large time-scale.

To explore further the common ground between sub-symbolic and iconic domains it is useful to gain more insight on musical objects – the smallest signifying particles of music. Their relationship to our perception of time is exceptionally important. Do we mainly experience signifying units as linked to time, or are time-independent properties perceptually more important? Another concern is the hierarchical nature of musical messages and musical communication. And yet another, very basic issue concerns the ontology of an object signified in a musical semiotic process: is it a static or a dynamic entity, and in what way(s) does it change or evolve in the course of a musical work? At the outset of this study, it was stated that our focus would be on patterns in time. Nevertheless, vertical structures in music remain worthy of analysis, though they do not number among our concerns at this time.

## 5.2 Legisigns and the temporal dimension

It has already been established that the manifestations of a musical object live a dynamic life cycle during a musical work or musical performance. It turns out that what we have so far been calling "musical objects" could easily be described in semiotic terms as *musical legisigns*, and surface-level instances of them could aptly be called *sinsigns*. In this case, a musical legisign could mean a number of different things. It might refer to traditional music-theoretical notions concerning melody, meter, rhythm, harmony, counterpoint, and the rules governing them. Also, legisigns reflect the commonly accepted practises in a certain musical culture or genre. Such notions could be called *general* or *cultural legisigns*. Yet another type of musical legisign is also conceivable, which derives from invariances that are valid within the context of a single musical work. We might call these *particular* or *embedded legisigns.* Invariances that reveal the existence of embedded legisigns may be found in characteristic motifs or thematic material. Nevertheless, it must be emphasized that themes or motifs in their individual appearances should not be considered as legisigns, for such appearances undoubtedly are instances, and as such, sinsigns. An embedded legisign sums up the typical properties or details of a certain musical idea, motif, or characteristic. In a way it is a (proto)type that can have many various instances or tokens – or even disguises – at the surface (sinsign) level. This distinction is quite crucial in the following parts of this study. Sometimes the legisign is manifested in a continuing process of expression that leads from a distinct starting point to an ending via a number of intermediate states. This is the *developmental tendency* in music. At other times, the object establishes a steadier form, manifesting its life in deviations from the latter. This is the *tendency to variation*. At still other times the object might at first produce a stable appearance, then generate new patterns that stray further from it, but finally return to the initial manifestation. This departure-and-return pattern might be called the *cyclic tendency*. These time-related tendencies, together with other conceivable ones, are the basic constituents of drama in music. They are qualities, and thus from the semiotic point of view can be treated as qualisigns.

In any of the qualisigns (tendencies) belonging to certain sinsign-legisign (instance-object) system-dynamics, whatever the qualisigns might be, it is quite clear that recorded music, regardless of the storage format, has only indirect access to embedded legisigns, i.e., to deeper levels of musical signification. A group of notes in a musical score is a sign that allows for a plethora of different interpretations, and that has one or more abstract legisigns lurking behind it. In contrast, a group of sounds in a recording reflects only one possible manifestation, one possible sinsign that implements or obeys its respective legisign(s). This type of manifestation is iconic, in the sense that it bears a resemblance to its object. But the object – the legisign or musical motif – becomes defined sufficiently only in the context of the complete piece. Thus, the entire piece functions as another sign, which interprets an individual moment. What its object really is, however, is not so evident. In fact, it signifies the complete musical message embedded in itself, both as a whole and all of its possible parts, just as a story refers introversively to all of its constituents. A sign in general represents its signified. But in musical contexts we often realize that the subject matter of music is simply music itself. A piece represents its own content, and for this reason may be called an *auto-icon*.

Auto-iconic meaning is of course not the only possible type of meaning in music. It is in a key position, though, because it is an essential part of meaning in almost any type of music. From the point of view of auto-iconicity, a musical object needs to be examined in the context of the whole musical work. Approaching legisigns behind sinsigns becomes possible only when one takes a holistic view, which construes the context of a musical continuum as a unity.

Earlier (Ch. 3.2) we discussed negative critiques of the idea of auto-iconicity, such as the one by Raymond Monelle. Perhaps now it is appropriate to take a look at Peirce's thoughts on the matter (2.230):

> ... in order that anything should be a Sign, it must represent, as we say, something else, called its *Object*, although the condition that a Sign must be other than its Object is perhaps arbitrary, since, if we insist upon it we must at least make an exception in the case of a Sign that is a part of a Sign. Thus nothing prevents the actor who acts a character in an historical drama from carrying as a theatrical property the very relic that that article is supposed merely to represent, such as the crucifix that Bulwer's Richelieu holds up with such effect in his defiance. On a map of an island laid down upon the soil of that island ... must, under all ordinary circumstances, be some position, some point, marked or not, that represents *qua* place on the map, the very same point *qua* place on the island.

From this statement, we may conclude that Peirce's own view of the matter was not altogether fixed. Yet he certainly leaves the door open to auto-iconic signification in the case of physical objects that signify themselves. Why, then, would it be any more difficult to consider the possibility of immaterial, time-dependent musical signs that signify themselves?

In a discussion on the emergence of meaning in a musical context, questions of hierarchy cannot be skirted. Almost any analytical system or idea dealing with musical form must also deal with the notion of multiple layers of information: surface layer, deep layer, and perhaps a multitude of intermediate layers. Any discussion in such terms could be characterized as Chomskyan, since the influence of Noam Chomsky's theory of generative grammars has been a dominant one in the humanities since the 1960s. However, I would take issue with the notion that all reasoning about surface and deep structures essentially derives from Chomsky's work. If we maintain that symbolic signification is based on iconic processes, we already have a hierarchy. Consequently, when we discuss musical objects and their various surface-level appearances, the discussion is not based on the point of view of generative grammars, but on that of sub-symbolism. Therefore, the claim is made here that *it is possible and justifiable to discuss deep structure and surface structure without making extensive reference to Chomskyan grammatical representations.* These notions in question are more universal, and need not be confined to language. Surface structure and deep structure can rightly be discussed as a dialectic between sinsign and legisign.

Empirical analysis mainly has direct access only to the surface structure of music – sinsigns acting as manifestations of musical legisigns – and only indirect access to the legisigns themselves. Hence, an analytical system is by definition one that is capable of approaching the signs that lie beneath the surface of musical phenomena, by producing generalizations of the dynamic surface of sinsigns. The generalizations explain the musical legisign by engaging with its properties and behavior, knowledge of which comes from studying surface manifestations in context. It is the context that can transport meanings to the receiver's side in the musical communication chain. The purpose of such study is to produce a generalized idea of the sinsigns. Consequently, a legisign is a generalized, abstracted product of sinsigns, which comes into being as a result of analysis. As generalization and abstraction take place, the sign necessarily loses part of its iconic nature, which gives way to symbolism.

The auto-iconic, sinsign/legisign network forms a special semiotic system that can produce meanings in a self-sufficient manner, and that allows for the emergence of abstract concepts without requiring any exogenous information about coding/decoding. Intuitively, however, it seems essential to define some ground in contrast to the sign. Before a discussion on the role of the ground, a closer look at the mechanisms of auto-iconic signification is needed.

## 5.3 On the emergence of new concepts

There is a striking similarity between the emergence of symbols from sub-symbols in connectionist system, and the emergence of symbols from icons in an auto-iconic process. The relationship between sub-symbolic and iconic domains of communication obtains by means of data-reduction techniques. We could say that the emergence of symbolic representations from a sub-symbolic information base is a parallel to the way symbols depend on icons. It would be a mistake, however, to conclude that sub-symbols and icons are one and the same thing. Instead, it is obvious that sub-symbolic signs simply possess rather strong iconic qualities. We have already noted that associative processes have an indexical aspect as well, whether the context is musical or some other kind. Sub-symbols are inherently iconic signs, and their dynamic behavior is very much like that of icons. It has already been suggested that iconism might serve as the basis of an analytic technological system. In the remaining chapters of this study, we shall investigate how successfully this might be done by using a particular parallel computing system.

A rather intriguing philosophical question concerns the parallelism between connectionist and semiotical processes of emerging structures. Based on what has been said thus far, it is obvious that symbolic information may be generated by a technical system. On the other hand, in semiotical systems, higher-level concepts may emerge from dynamic processes among elements of lower-level information. Consequently, one might reach the almost frightening conclusion that abstract concepts could originate from a strictly technological procedure. The experiment described in the following chapters provide additional information, which can help us decide if such a conclusion is valid.

## 5.4 Musical signification as a set of procedures

The marriage of semiotic and computational concepts is performed here with formalisms that use a step-by-step model and apply it toward an analytical end. We call the application of such formalism *procedurization.* The latter term seems has a strong association with algorithmic ways of problem-solving. The use of algorithms for analytical purposes is not unheard of in musical semiotics. Indeed, one branch of semiotics seeks to formalize analysis almost to the point of algorithmic form. This branch stems from linguistics, and may be exemplified by Nicolas Ruwet's music-analytic method. We shall refer to Monelle (1992: 83-88) for an account of Ruwet's mechanism of discovery.

Ruwet's "analysis algorithm" consists of four main steps. At the first stage one studies the material in order to find the longest recurrent passages. At the second stage one identifies the non-recurrent passages. The third stage consists of a comparison of length between recurrent and non-recurrent passages. At the final stage one compares all passages in order to find the ones that are variants of the same passage – or in the terms we have used here, the ones that are superficially different instances of the same musical object.

In this study, however, we do not insist on rigorous algorithmic formality, such that algorithmic formalisms are considered synonymous with symbolic data-processing. The procedurization of ideas can also take place in a connectionist system, which is not an algorithmic procedure in the strict sense. Thus, here procedurization does not imply that a strict algorithm or grammatical rule-set is the main objective, but rather that the problem should be structured into subparts and tendencies. Processing of the problem should be taken to the point where it can be subjected to computation using either a deterministic algorithm, fuzzy logic, or a connectionist network.

A graph representing various factors and mechanisms of musical signification is introduced below. It depicts four progressive stages. First the graph will display the four agents of signification: object, representamen, interpretant, and ground. In addition, two relations are shown. The first relation is between an object and its representamen, and is called *instantiation*, or the making of an instance (Peirce 2.230).[10] This relation reflects the way representamens are generated from an object. The purpose of this procedure is to show that a single musical object may have a number of instances. In a way, a musical object is a class, an idea, a collection of attributes, or an abstract entity. Correspondingly, instances, or representamens, are actual-world realizations of the respective class or idea. Upon instantiation, the options allowed by the model object ("class object") are fixed, so as to bring about its realization as an individual sinsign.

The second relation is the one between representamen and interpretant. Peirce calls this relation *interpretation*: "A sign ... addresses somebody, that is, creates in the mind of that person an equivalent sign, or perhaps a more developed sign" (Peirce 2.228). Interpretation produces a new sign, which is an initial step on the way to associative chains. Interpretation is often immanent, which is to say that a musical interpretant is usually a product of the listener's mind. Alongside this type of immanent interpretant, Greenlee (1973) recognizes other interpretative signs. In the context of linguistic signs, such interpretants may include a written translation, a spoken sentence, and the like (Monelle 1992: 194). In computer analysis of musical data-streams, various representations of musical signs may function as interpretants, whatever their explanatory value might be. As the first of a series of step-by-step diagrams of signification processes, Figure 5.1 depicts mechanisms of interpretation and instantiation.

Since both representamen and interpretant are signs, it is possible to think of another semiosis, in which the interpretant as a representamen, and produces another interpretant. This may in turn act as representamen. In this way, a *chain* of semiosis can be iteratively formed.

Monelle refers to the "infinite iteration" of the interpretation process, and attributes the idea to Gilles-Gaston Granger (1968: 114) and Douglas Greenlee (1973: 26).

> This suggests an infinite regression; if the interpretant is itself a sign, then in its turn it will need an interpretant. The road sign is interpreted by an official sentence; this verbal explanation is

[10] Incidentally, the term *instance*, present in Peirce's text, has been put to frequent use in the technical context of computer science. It usually refers to a technique in object-oriented programming (OOP), which uses a categorization surprisingly like the process explained above, in reference to legisigns and sinsigns; however, these notions are replaced by the terms *class* and *instance* in the programming jargon.

significant by virtue of a greater sign, the whole system of the highway code. This interpretant will need an interpretant, and so on ad infinitum. (Monelle 1992: 194)

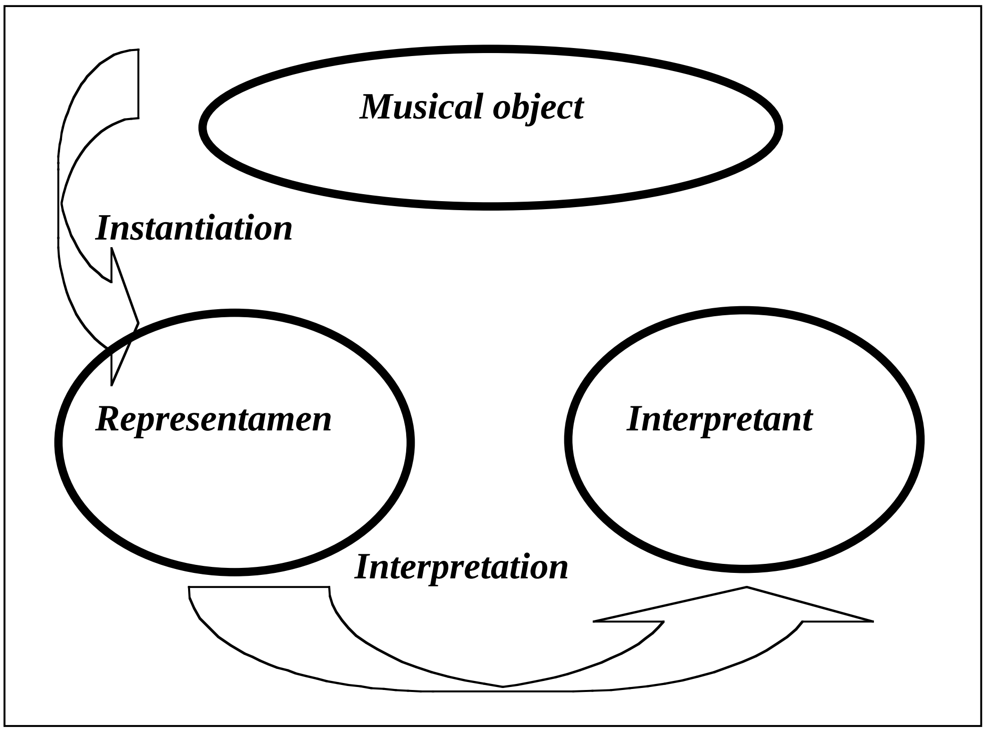


**Figure 5.1** Instantiation and interpretation

José Luiz Martinez discusses the process of interpretation in depth. After suggesting a number of different kinds of interpretation, and partly following another triad used by Peirce[11], he states that the step-by-step process from immediate interpretant to final interpretant is (at least in a musical context) not necessarily a simple journey by way of the shortest possible route between two points:

[11] The Peircean triad in question comprises emotional, energetic, and logical interpretation. Because it is not a directly integral part of his famous categorization of signs, it is less frequently used and cited than are the triads concerning sign categories.

> The utterer and the interpreter are not simply the musician(s) and the listeners; and, above all, these concepts must not be understood as engaged in the simplistic, indication scheme of sender-message-receiver. The utterer and the interpreter, caught up in the web of semiosis, may even be difficult to locate in some circumstances, for they are thoughts among thoughts among thoughts.... Besides, in many musical situations their positions are irrelevant. (Martinez 1997: 78)

In other words, the semiotic process holds the possibility of a kind of polymerization. How far polymerization, as such, might in principle continue also depends on the kind of interpretation in question. Obviously, the interpretation of sinsigns may produce legisigns, in which case a final interpretation will probably be arrived at, sooner or later. However, it is also a conceivable that an interpretation of a sinsign would be homogenous, in the sense that it might produce the same type of sign as a result – another sinsign. In that case, a final interpretation may be much further away. It is even possible that the final interpretation does not exist at all.

It is possible to track down the constituents of such a polymer – the constituents being transformations and reinterpretations of the musical idea at the level of sinsigns, along the span of the musical work. This is reminiscent of a number of linking or looping processes, which have been frequently used in computer music theory and in simulations of composing strategies. Markov chains are perhaps the most well-known example of such algorithms (e.g., Xenakis 1971: 43-78). Another, somewhat similar case would be the idea of "grammatical chains", as discussed together with context-sensitive grammars by Kohonen et al. (1991: 231-235). The motivation for the chain metaphor of a grammatical rule-set was the desire to mimic or approximate cognitive mechanisms of association and memory.[12]

The case of chaining, or polymerization, in a semiotic process differs from the Markovian and grammatical models in at least one respect. Because Peirce's idea of interpretation was conceived as a psychological and cognitive mechanism, instead of a strict algorithm, the distancing from initial

[12] The term "grammatical chain" is my own, and is not used in the publication by Kohonen et al. In private communication Kohonen has emphasized that the special grammar used in his experiment is not as such related to Markovian processes. Nevertheless, in the experiment described by Kohonen, his grammar-based system was used in a manner similar to the Markov chains in Iannis Xenakis's well-known usage of them in musical composition.

interpretations by forgetting is hardly ever complete. If the system of object-representamen-interpretant is considered a genuine trichotomy, as Peirce argues, then some distinguishable “background” presence from part of the original object will certainly appear, even in later links of the chain. In other words, the cognitive memory mechanism is fine-grained, preserving selective reminiscences of the original situation(s) for long periods of time. Memory links may be parallel and tangled, so there will be links at several points in the chain. The topology of the model implies that more than one dimension is necessary. Consequently, where semiotic relationships are concerned, the linear, chain metaphor should be replaced by the network metaphor. Because interpretation-reinterpretation relationships constitute a framework for the concept of association, it follows that the representamen-interpretant network is profoundly involved with the technology of associative systems and artificial neural networks, inasmuch as technical implementations are concerned. The connection between interpretation and association was referred to by Peirce, in the context of rhetoric as a way of approaching the interpretation of concepts: “the task [of pure rhetoric] is to ascertain the laws by which in every scientific intelligence one sign gives birth to another, and especially one thought brings forth another” (2.229).

In constructing a computer model, however, it is necessary to keep things simple. Thus, it is not possible here to emulate associative recollections of a complexity comparable to those carried out by true cognitive mechanisms. Instead, in using a simple computational system, we shall approach cognitive processes by way of procedures oriented toward the more limited phenomena of short-term memory.

Before we proceed deeper into our signification model, a final remark should be made concerning data-representation. In the following, we shall largely avoid discussions about formats in which music should be represented. The format used may be either a direct translation of acoustic phenomena into a digital form such as sound (sample) files, or into one of the many available score-oriented or MIDI-oriented formats. We simply presuppose the existence of a signal consisting of an ordered set of discrete, iconic signs in time.

## 5.5 Mechanisms of instantiation and the idea of semiotic ground

Instantiation, which plays a major role in our model of music reception, may be characterized as the translation of a deep-level entity, such as a legisign or musical object, into a surface-level entity, i.e., a sinsign or representamen. An object can generate a great variety of representamens – one legisign may be instantiated such that it produces many different instances. The number of possible instances could even be infinite. On the other hand, sinsigns of common appearance could in some cases be instances of different legisigns. In such cases, the correct meaning would be determined by the context. In Peirce's writings, attention is given to this fact. According to him, instances of similar appearance but different significance can be considered as discrete parallel meanings (different legisigns) created by a single sinsign instance, as they are manifested from the point of view of a sinsign:

> The word Sign will be used to denote an Object perceptible, or only imaginable, or even unimaginable in one sense – for the word "*fast*", which is a Sign, is not imaginable, since it is not *this word itself* that can be set down on paper or pronounced, but only *an instance* of it, and since it is the very same word when it is written as it is when it is pronounced, but is one word when it means "rapidly" and quite another when it means "immovable", and a third when it refers to abstinence. (Peirce 2.230)

What Peirce says above applies to a linguistic context, in which meaning is mainly symbolic and parallel meanings remain relatively discrete, that is, separate from the primary meaning. Musical meaning, on the contrary, largely resides in iconic semiosis, and therefore allows for a wide, continuous spectrum of somewhat indistinct or representamens. Therefore, in musical meaning, signs of similar appearance but differing signification will be more apt to evoke associations, and hence more apt to take an active role in forming connotations and building structural cues that contribute to the *embodied* *meaning* of the message.

How sinsigns are formed on the basis of legisigns, with respect to qualisigns, is a question of importance to music analysis, whether the process is called instantiation, *Auskomponierung*, or something else. Peirce (2.228) maintains that it is the dialectic between *object* and *ground* that, through instantiation, brings forth the representamens. We have already developed a notion of what the object – the musical legisign – in the limited context of one musical work might be. It is an idea behind sinsigns, which is to some extent distanced from the details, yet close enough to musical reality that it can be expressed in such terms that it can easily be realized as legisigns. The object has some abstract properties, forms, and/or attributes that are projected in music and that stand out in respect to the ground. This fact returns us to the concept of ground in musical semiosis.

Two statements sum up the view of ground assumed in this study. First, the musical ground is not completely arbitrary – it is not made up of symbolic knowledge alone, even though it expresses cultural conventions. The ground is not arbitrary because of the iconic character of musical semiosis. An iconic sign conveys information by relationships and proportions, and the ground plays an active role in defining these proportions. Second, we must distinguish between different types of grounds of musical sinsigns – cognition-oriented, culture-oriented, genre-oriented, and work/piece/performance-oriented grounds, from the most general to the most particular. Each different contextual ground takes its own attitude toward iconic and symbolic modes of signification, a fact which deserves further explanation.

Components of the *cognition-oriented* ground have their basis in extra-musical reality. This has a pronounced effect on such things rhythmic patterns, which connect to psychological laws of perception. Rhythms and tempos, though largely matters of culture, are also tied to our bodies' biological clocks. Our ideas about harmony are built on the natural system of wave-forms, harmonic series, partial tones, and the mechanisms of hearing such things. These have important iconic relationships with physical and physiological frameworks on which cognitive processes are based, such as the way sound is conveyed to our nervous system through the cochlear transform of physical vibration into neural signals.

The *culture/system-oriented* aspect of ground involves laws of a prevailing musical system, such as tuning lattices, rules of harmonic thinking, scales and keys available, rhythmic patterns, tempos, timbres, and properties of instruments.  It involves traditions, both written and aural, concerning composition and performance.  All this forms a body of knowledge that makes up a musical culture, and involves less iconic aspects of signification than does cognition-oriented signification.

The *genre/corpus-oriented* ground involves conditions and characteristic features determined by musical style.  This ground is more conventional than are the mentioned general characteristics of entire musical cultures, but may sometimes possess iconisms having to do with extra-musical aspects of culture, such as social codes for courtship or interaction between sexes, as in the case of dance music.  Many norms of genre-oriented musical signs are qualisigns.

The *work/piece/performance-oriented* ground may play a larger role in the domain of art music than it does in popular, dance, and ritual music, because art music tends to strive more for originality and continued expansion of its vocabulary.  This ground permits us to approach the dynamics of musical sinsigns and legisigns at the level of an individual musical work. Iconicity is an integral part of the dynamics that produce meanings at this level of signification, for it lends itself handily and understandably to the interaction of signs within a musical work.  Into this niche fits our strategy for algorithmic analysis of music-based time series.  In what follows, the term *ground* refers primarily to this work/piece/performance-oriented ground, which, contrary to other types of ground, is well-suited for a reasonably simple computing environment.

In general, the ground provides a framework for interpretation that leads from representamen to interpretant.  Considering that the ground is so multi-layered and multi-faceted, we appreciate the fact that the relations between it and between both representamen and interpretant is a complex process.  To lump such complexity under a single rubric, and use it as a semiotic operand, is quite a reduction indeed, but may be useful as a theoretical aid.  The composite role of the semiotic ground is illustrated in Figure 5.2.

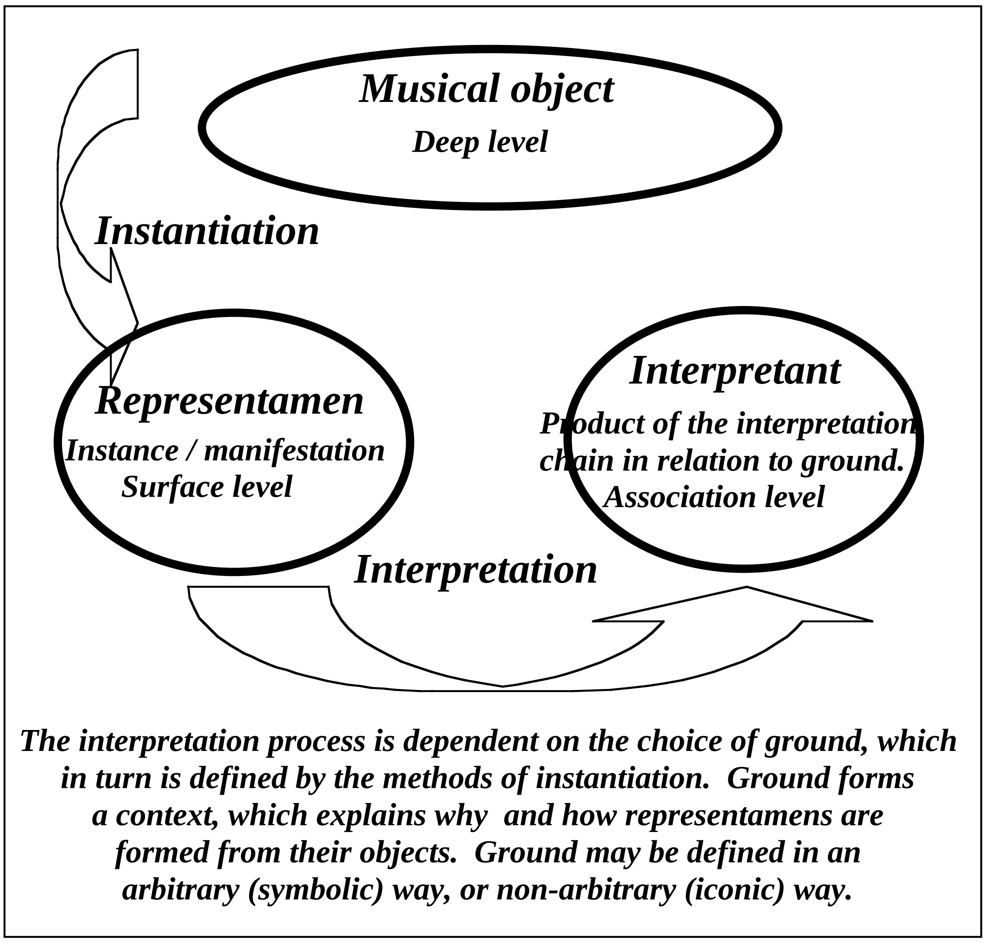

Musical object
Deep level
Instantiation
Representamen
Instance / manifestation
Surface level
Interpretant
Product of the interpretation chain in relation to ground.
Association level
Interpretation
The interpretation process is dependent on the choice of ground, which in turn is defined by the methods of instantiation. Ground forms a context, which explains why and how representamens are formed from their objects. Ground may be defined in an arbitrary (symbolic) way, or non-arbitrary (iconic) way.

**Figure 5.2:** Ground, representamen, and interpretant

Let us now turn to a somewhat novel idea of the interpretant in music. In music semiotics, probably the most common notion of interpretation is the one that emphasizes symbolic signification in the relationship between sinsigns and their ground. In such a case, we are dealing cultural and genre-oriented interpretation through an existing person. According to Peirce, an interpretant is generated when a "sign ... addresses somebody" (Peirce 2.228), meaning that semiosis involves someone who receives a sign. The interpretant lives in the listener's mind. A question then arises: What about more iconic, less subjective signification processes?

Using the work/piece/performance-oriented ground would make it possible to establish an interpretation process *within* a musical piece or performance. The point of view becomes that of the transmitting side of musical communication, and the focus is on the internal message of the transmission – i.e., features in the signal which can be used to decode the message without need for an external “Rosetta stone”. Associations in the signal become the key elements in decoding the message. This is how the representamen-interpretant, interpretation-reinterpretation network directly manifests itself in music. The associations are established in a networking process inside the musical message. In this mini-universe, the musical intelligence and competence of composer and performer serve as engines and generators. In a somewhat strange way, this adds a slightly impersonal flavor to the semiosis, since the whole process is expressed only within a musical work. Nevertheless, a source of musical intelligence is needed, even though it works in such a restricted domain as that of a single piece of music.

## 5.6 Reverse instantiation and association tracing

“Reverse composition” is necessary in order to get at information about a musical work. One needs to have a specific manifestation or group of manifestations (sinsigns) as a starting point, and proceed towards a general definition of the musical legisigns in question. Since we have defined both instantiation and interpretation as taking place in the piece, reverse operations are needed; namely, reverse instantiation, or *deinstantiation* (an equally appropriate name might be *generalization*), and reverse interpretation, or *association tracing*. Without going in to detail about how such operations might work, their positions and relationships are shown, together with the operands and operations of earlier diagrams, in Figure 5.3.

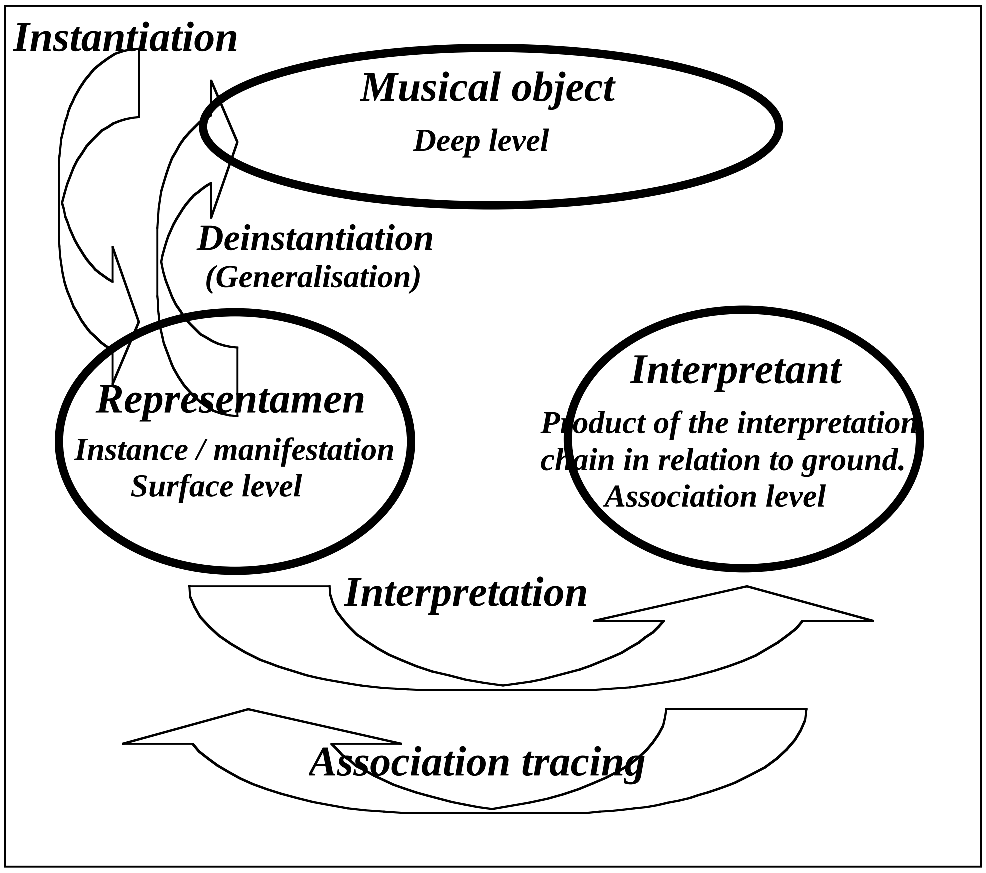


**Figure 5.3:** Association-tracing

The starting point for analysis of a musical signal is first to consider the data, all the given signs, as sinsigns. Initially, there is no way of determining whether two or more sinsigns are manifestations of a common legisign, or which sinsign would be the most simple materialization of the legisign in question. This means that, at first, all manifestations of the musical object are equally important. No sinsigns are to be taken as having a primary or preferable form over the other ones, since the distinction between representamen and interpretant in relation to ground has not yet been made clear. As stated above, by ground we refer primarily to the work/piece/performance-oriented type of ground, which is the most feasible concept of ground in the context of algorithm-driven analysis.

A process giving rise to signification in the reception of musical signals is sketched in Figure 5.4. The figure also indicates the multi-dimensional nature of the legisign/sinsign[13] process of semiotic interpretation and reinterpretation. Chains thus formed may, in conceivable practical situations, have considerable length and a high degree of connectedness. The graph represented in the figure may be called a *reinterpretation network*.

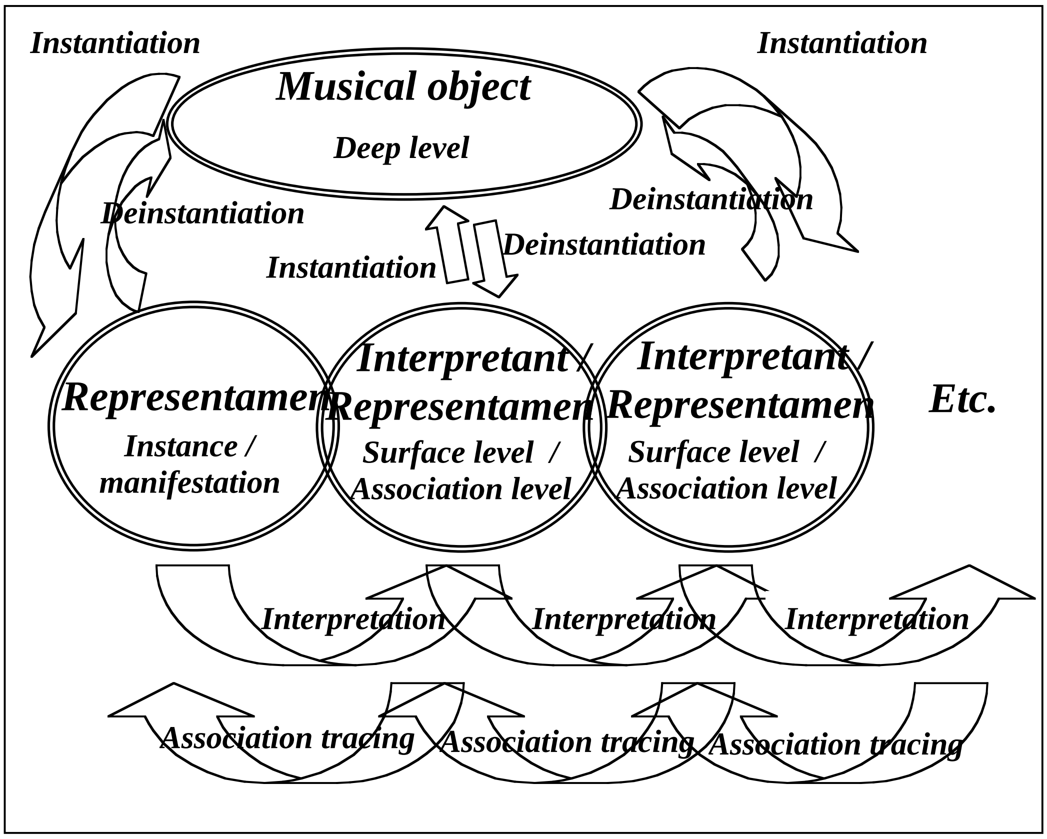


**Figure 5.4:** The reinterpretation network

Each of the sinsigns in a musical signal is a node of the reinterpretation network. Each node has a logical link to the corresponding legisign. Information about the musical object is not encoded into any single one of these object-representamen arcs, but to all the nodes and arcs as a whole. This

[13] Throughout the present text, "legisign/sinsign process" is a term used synonymously with that of surface level/deep level process or object/manifestation process.

is a natural link to connectionist thought, and an argument for using technical solutions of a similar nature in order to engage the problem. The network forms a distributed representation of data. Generalizing properties of a suitable connectionist network should provide access to attributes of the legisign, and thus carry out deinstantiation. Obviously, cognitive reasons also speak for the connectionist approach: in musical signification, intuitive, non-formalized knowledge usually plays a crucial role where instantiation or meaning in general is concerned. Since explicit rules for instantiation are usually not available, neither are rules for deinstantiation.

The role of the work/piece/performance-oriented ground is a complex matter because of the multiplicity of legisigns in a single musical signal. One challenging feature in association-tracing is that in a single semiotic space created by a time-ordered data sequence, there are a number of legisigns, and the corresponding sinsigns for each of them can exist throughout the sequence, in variable quantities and even overlapping. Pattern recognition involves three problems:

1) how to differentiate between ground and legisign – i.e., how to distinguish between those signs in the signal which manifest the ground, and those which manifest a legisign;

2) how to distinguish among signs manifesting different legisigns;

3) how to find all the manifestations of a certain legisign.

Little can be known about the possible number of either legisigns or sinsigns in a free data-sequence, if no limitations are specified concerning the kind of musical material that can be processed. Nor is it easy to give any information, concerning their general properties, that would facilitate algorithmic processing. On the other hand, if such information could be specified, then there would be extremely few possibilities for extracting relevant and interesting information about music, for music that yields to such limitations would likely be banal at best. Systems aimed at taxonomic classification of a free sequence must be able to find the ideal number of classes, and to distinguish between sinsigns and ground. This may be an idealistic position, though. Perhaps elementary information about the ground will always be needed, at least at the level of what Hofstadter calls the "frame message (1997: 166). The latter refers to the capacity of messages to point out that they actually are messages, i.e., those traits that say, I am a sign. In any case, keeping the ground information to a minimum will assist us in maintaining the general motivation and applicability of the method.

Conditions like the above have not been easy to reach with conventional computer programming methods. If they can be fulfilled, then deinstantiation may take place.

Provided that a procedure for deinstantiation is available, the next step would be to construct a model that reverses interpretation, by reverse tracking of the associative references of the time sequence. *Association tracing* (AT) uses knowledge provided by the deinstantiation process – as we have seen, deinstantiation is a prerequisite for AT. The latter provides new information about the main structural units of the time series. The beginnings and endings of structural segments will stand out, and appropriate segmentation may be studied by means of the logical structure provided by the data. In this way, groundwork may be laid for mapping the tendencies of the piece, be they developmental, variational, or cyclical. If successful, such a treatment will reveal the general direction of the interpretation chain.[14]

In unsupervised learning the grounds for classification and recognition of patterns in the input signal are data-driven – they are retrieved from the signal itself. Flexibility and generality are desirable traits of any analysis system. If classification of sequences may be carried out in a data-driven manner, it is easily imaginable that interpretation may also be accomplished. In Figure 5.4 a scheme was introduced that presented interpretation in musical semiosis as a process of connecting to a new sinsign from a previous one. If we thus take interpretation of a musical sinsign as interaction that takes place mainly among sinsigns, then the process involves sub-symbolic representations of data. In the sinsign-legisign network of the music, surface-level entities communicate with the deep level through sub-symbolic information. The relationship between sub-symbolic and symbolic representations is not a strict, unidirectional hierarchy, but a tangled one on the order of Hofstadter's heterarchies (discussed in Chap. 4.4).

[14] The appropriateness of the chain metaphor has already been questioned in this context. After all, our treatment of interpretation depends on a parallel networking process. It is used here only because a musical time-series is a sequential medium. In that sense, it is a one-dimensional string, even though it may possess a plethora of dimensions, which can be revealed by studying its structure in the light of music theory, as well as from the point of view of its signification.

The semiotic status of these sub-symbols can now be made more specific. Connectionist semiosis is characteristically sub-symbolic, but these sub-symbols are a special case of iconic sinsigns, not generic icons. The sub-symbolic transform maintains a *diagrammatic* relationship to their objects, which is to say that reversibility of process is a significant quality for any such signs. For example, a series of sub-symbols may be constructed to reflect the pitch structure of a melody, and the pitch structure in question should be retrievable from the series. This reversible nature of data set and its object is a distinctive feature of a sub-symbolic treatment in its pure form. A diagrammatic relationship between a sign and its object can be conceived not only in semiotic terms but also in mathematical ones – it is a *monotonic function*. This situation should not be confused with the relationship between two sinsigns in the sign-object-interpretant schema, which does not need to be diagrammatic any more than it needs to be monotonic. The creative interpretation process in musical signals has considerably greater freedom.

# 6 Summary of principles used as the basis for the experiment

To draw together the ideas presented in the previous chapters, and as a starting point for the empirical part of this study, we now present the working hypotheses for the experiment:

1) The aim of the experiment is to create a model for segmenting time-sequences in a flexible, simple, and compact form. The functionality of this aim is tested by means of a computer program that implements the model.

2) Flexibility, in this context, means applicability to a variety of musical styles, and the ability of the system to handle signals complex enough to exemplify real-world musical situations. Simplicity and compactness add to unity of concept. In practice, this means that the computer program, which models the theory of analysis, reflects a single, clearly definable principle of operation.

3) The experiment can to some extent be evaluated in terms of cognitive justification. Nevertheless, the main goal is not to create a new cognitive model. Instead, the purpose is to build a functional performance model of a single process with no unnecessary theoretical ballast, even though the relationship of the core idea to various cognitive models deserves consideration.

4) The traditional way of building a performance model capable of analysis has included a model of the listener's subjective knowledge. To include such

information is practically equivalent to claiming that a performance model should always contain much *a priori* musical competence. On the other hand, one may suppose the contrary – that the inner cohesive force of a musical work contains enough structuring information on which to base an analytical process. In this case, the necessary competence would be drawn from the work being analyzed. This is made possible by the interaction of various parts of a musical passage, this interaction constituting the internal cohesive forces of music. The latter strategy seems more interesting from a systems-theory point of view, since the operation of the system is triggered by music.

5) In the experiment we pursue quantitative aspects of the foundations of musical meaning. It is somewhat paradoxical that a system applicable to data sequences in general should be "more musical" than those which include *a priori* knowledge of music theory. Yet one can find reasons for such a position in Leonard B. Meyer's idea of *embodied meaning* in music. As Meyer states, an important part of meaning in music derives from "stimuli pointing not to extra-musical concepts and objects but other musical events that are about to happen" (1956: 35). Embodied meaning belongs to the internal reference structure of a piece. It involves detection of similar and different subsequences within a sequence, in such a way that covers different degrees of similarity; such detection goes beyond merely checking for identical elements. Such an idea of similarity/difference is believed to be the bottom line in thinking about musical form. Whether or not references to other musical works or to extra-musical objects occur in a piece (which would manifest the presence of culture/system-oriented or genre/corpus-oriented grounds of semiosis), the internal reference structure acts as a unifying force. Since our experiment strives to exclude extrinsic musical concepts, we naturally turn to embodied meaning as a framework for our ideas.

6) The experiment takes the massive-parallel approach used in cognitive and computer sciences. A massive parallel system can operate mainly with or without *a priori* knowledge. Some *a priori* knowledge will in any case be present, for signification necessarily involves the cultural/system-oriented ground in respect to the sign at some level.

7) In sum, the present study aims to use minimal *a priori* knowledge in a sub-symbolic performance model of iconic sinsigns and their processing, for

the purpose of segmenting time-ordered data sequences that originating from musical signals.

By assuming these working hypotheses, we face a number of opposing views in the literature. To conclude his discussion of segmentation strategies of music, Monelle states the following: "Segmentation in music will always be ultimately based on intuition, because the relation of phonology and semantics, of expression and content, functions differently in music ..." (1992: 89).

This study is in part an attempt to challenge Monelle's statement, though he ultimately acknowledges the need for formality: "Analytical segmentation should be based on rational and explicit principles" (ibid.). Our experiment serves as a defense of using the work/piece/performance-oriented ground of semiosis as a basis for interpretation, as will be made clear in the following chapter.

James Tenney and Larry Polansky (1987) have produced a study similar to our own. The basic principles differ, however, in that the authors rely on rules derived from common Gestalt laws (proximity, similarity of parameters, mean interval, boundary interval). It was considered desirable here to use *pattern* similarity – likeness and difference as the defining structural factors – instead of *parameter* similarity. Our aim has been to avoid dependence on any prefixed properties of phrase or pattern at the possible cost of generality. In contrast, Tenney and Polansky's adaptation of Gestalt laws seems to be influenced by classical notions of what melodies should be like. This leads them into difficulties, which are dealt with by the introduction of weight parameters into the system. The problem still remains of not knowing in advance what the weights should be or how they should be determined, and this leads to a trial-and-error way of problem-solving. While we share some of their views, their terminology is different from the one used here. For instance, by *sequence* they mean a particular hierarchical level of temporal organization consisting of *clangs*, which in turn consist of *elements*, whereas here the term "sequence" is used rather interchangeably with the more general (computing) concept of a *string* of elements.

Another study that is closely related to our own is the one by Emilios Cambouropoulos (1996a, 1996b). It, too, has a different focus from that of the present study, in that it is constructed on common Gestalt laws, as was Tenney and Polansky's earlier work.

# 7. Test-report on an association-tracing system propelled by self-organizing feature maps

## 7.1 Features of software implementation

We now put to use ideas discussed earlier, about the machine model of learning, or adaptation to data environment. The operating strategy for the machine is one of unsupervised knowledge-acquisition. The particular operating principle is a self-organizing map (SOM) based on an altered version of Kohonen’s model (see Chap. 4.8). Hereinafter, the modified Kohonen model will be called SOMAT (Self-Organizing Map with Association-Tracing).

Current musical applications of SOM method are often designed for classifying the organization of time-independent, non-sequential elements (Leman 1991b). In contrast, the present task involves segmenting a time-sequence, in order to detect the possible phrase structure in it. It is first necessary to explain further the nature of this data stream.

*Time sequence* refers here to a string of “equi-durational” data elements obtained by the sampling of some real-world process at constant time intervals, and presenting it as a sequence of values. Each value-element represents a quantity depicting the process. If only one variable quantity is sampled, there will be only one data value attached to each element, and we can call the sequence one-dimensional. One could easily think of cases wherein several quantities, constituting a multi-dimensional vector, would accompany each sampling point in time. We do not attempt to tackle such cases in this study.

Much has been said, above, about the iconic nature of suitable input data for the SOMAT system. Accordingly, it seems likely that SOM method is not able to grasp the meanings in an arbitrary (symbolic) data stream. Rather, Kohonen's architecture is designed to reflect the relationships of given data in the end result, the map. These relationships denote a correspondence between data and their mapping in a non-arbitrary, iconic manner. Iconicity means that the data will not be processed by means of a mainly symbolic strategy. Moreover, it means that some internal organization exists in the data sequence itself, which reflects its represented object. This naturally requires that the object be a real-world construct (in this case, a work of art), in which some degree of cohesion and internal laws of organization exist.

In the present application, the data sequence is drawn from a musical score. As has been pointed out, a score of common-practice musical notation is iconic, to the extent that it represents musical pitch and time. The numerical representation maintains this iconicity, and even makes it more diagrammatic and reversible. A pre-process used for encoding the data samples the musical pitch at regular intervals, so as to form the individual signs in the discrete signal.

Let the time sequence (discrete signal) thus sampled be called $S$, and the length of the sequence be called $K$. Each sign $S[k]$ of equal time-span represents the pitch of a note at a specified moment of musical time. An $N$-dimensional feature vector $F$ consists of $N$ successive elements of this time-series. Each feature vector may be written as follows:

$$F = \{F[1], F[2], \ldots, F[n]\}$$

which is equivalent to:

$$F = \{S[k + 1], S[k + 2], \ldots, S[k + n]\}$$

For each location of the data sequence (save for the $n - 1$ last ones) $S[1]$, $S[2]$, …, $S[k - n]$, a feature vector of length $N$ can be defined. Such a vector essentially opens a window, $N$ elements wide, on the data stream. During each iteration of the teaching process, this window slides one element at a time, from the beginning to the end of the musical passage. A data stream of $K$ elements will thus be represented, step by step, with $K - N + 1$ feature vectors.

Since the feature vectors are introduced to the SOM one by one, each vector acts as a stimulus, prompting the map for a response. The map manifests its response by focusing on a cluster of activation centered around a single node at a certain location on the map. Feature vectors with greater mutual resemblance will typically prompt responses from nodes located close to each other, whereas feature vectors with greater shape-wise deviation tend to settle at distant locations on the map. This is in accordance with the topological ordering principle discussed in Chapter 4. Generally, the location of the response-focus reflects the ordering of knowledge on the map.

The representation of successive moments in a sliding time-window as vector components differs somewhat from other existing SOM applications, such as those of Kohonen (1989: 140-142) and Leman (1991b: 10). Hence, the SOM is here being put to a rather novel use in being applied to time-ordered vector components. Moreover, here the SOM acts as a measure of pattern distance rather than as a classifier.

Our initial goal, however, was not to produce new measurements of pattern distance, but to pursue the foundations of musical meaning. As stated earlier, we adopt Meyer's concept of "embodied meaning" in music, and the experiment seeks to establish such meaning in a computational way. In other words, the model concentrates on (self)referentiality within pieces of music. A semiotic view of association-tracing was introduced earlier. Let us now consider one possible implementation of it, in the form of "reference tracing".

Our working definition of "reference" is as follows. In the model, feature $F_a$ is considered to have reference to feature $F_b$ if there exists similarity between $F_a$ and $F_b$ according to some metric. The metric suitable for this purpose calls for further discussion. First, however, we shall define the tracing procedure itself, and merely suppose that there is a method at hand for determining the degree of similarity among the feature vectors. Let us further suppose, that there is an implementation of this method, which we call `ComputeSimilarity()`.

The tracing of similarities and differences among subsequences in a supersequence comes down to pattern-matching that is governed by some higher-level construct. Comparison of every possible subsequence against every possible subsequence is essentially a two-dimensional matrix comparison, which is simple enough in principle, though costly in terms of computing time; but even that will not represent the important information concerning segmentation. Instead of simple comparison, the model runs an iterative check procedure on preceding elements in the sequence from among the elements to be compared, in order to trace the starting points for correlation chains. Let the data sequence be called *S* and the output signal from the system be called *G*. We can now define two feature vectors, $F_a$ and $F_b$, at locations *a* and *b* on *S*. The correlation between $F_a$ and $F_b$ will be called $C_{ab}$, while $C_{tot}$ represents the running total of correlation detected in a chain of successive locations on *S*. A procedure for one step of reference-tracing is implemented roughly along the lines of the following, simplified pseudo-language example:

```
procedure LocateTransient (a, b, S, G)
begin
     $C_{ab}$ = ComputeSimilarity (a, b, S, G) // compute reference value
     if  $C_{ab}$  >  threshold                // if reference detected
          $C_{tot}$  = $C_{tot}$ + $C_{ab}$;            // increment total correlation
          LocateTransient (a-1, b-1, S, G);
                                        // the above line takes care of the backstep
                                        // by recursion with decremented indexes
     else
     $C_{tot}$  = $C_{tot}$ + $C_{ab}$;                 // increment total correlation
          $G_a$ = $C_{tot}$;                        // set output signal to $C_{tot}$
          return();
     endif
end
```

A less formal explanation would be the following. When a reference is detected between feature vectors in the data stream, the feature vectors $F_{a-1}$ and $F_{b-1}$ are checked against each other. If they also correlate, the same check is carried over to $F_{a-2}$ and $F_{b-2}$. The iteration is carried out backwards from locations $S[a]$ and $S[b]$; back-steps are taken until no more referentiality is found after *m* steps backwards. The locations $a - m$ and $b - m$ are marked; these correspond to the last feature vectors that correlate. It is likely that some locations in the sequence *S* attract such markings significantly more than other ones do. These are obvious candidates for transitory regions, and are treated as segment boundaries. The process is repeated as a comparison against every location $S[n]$ ($n = 1, 2, 3, \ldots, k$), where *k* is the total length of the sequence.

This tracing part of the algorithm, performed in a loop, may be recursive, as in the above example, or it may be iterative.[15] The markings generated in the process are collected in each location *n* to form the output signal *G*[*n*] from the input *S*[*n*], since the whole cross-tabulation is carried out for every *n*.

A measure of pattern similarity is needed to carry out the comparison. This part of the algorithm is carried out by another procedure, `ComputeSimilarity()`. A number of different similarity-measurement engines might be used for this purpose. Some earlier work of the present author includes attempts to use grammatical and computational constructs as such an engine (Tiits 1992: 348). Such attempts were found to lack sufficient flexibility for the analysis of free stochastic sequences. In the present project, the adaptive properties of the SOM serve as the pattern-matcher. The purpose is to provide the kind of flexibility necessary for processing real-world data, which does not always follow preconceived categorizations in an expected way. Evidence from the experiment seems to suggest that SOMs are able to make generalizations of musical patterns, i.e., to develop recognition abilities of general pattern-classes such as the ones depicted in Figure 7.1. All the patterns represent different prototypes of possible outlines for 3-component feature-vectors in terms of increasing/decreasing tendencies.

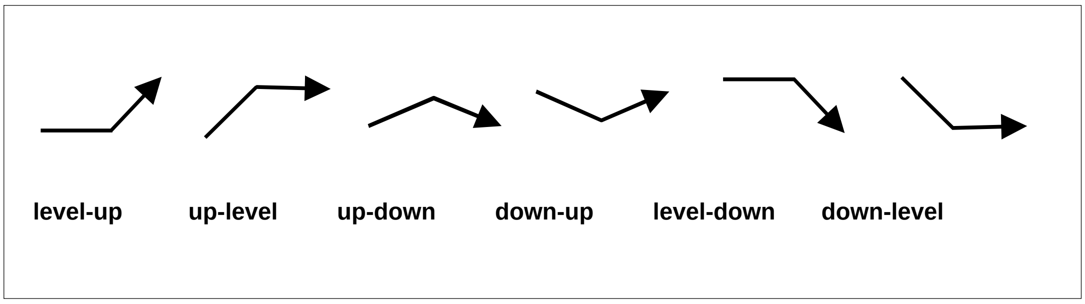


**Figure 7.1:** Some possible prototypes of pattern-classes

Though the SOM is a conceptually elegant method of categorization, most implementations of it involve heavy computation.[16] For example,

[15] More detailed documentation of these details of the SOMAT system is found in Chapter 8.3.1, and the complete listing for the program implementation is provided as an appendix.
[16] An SOM would naturally be effective in a massively parallel computing system, even in an analog computing environment. It is the implementation in digital serial machines that brings efficiency issues into the picture as being potentially problematic.

Schalkoff (1992: 284) estimates the number of teaching cycles as typically being around 10 000 or more, so as to have self-organization take place at a desirable level of accuracy. It should be mentioned, however, that documented experiments have been conducted with less than 1000 teaching cycles. Kohonen (1989: 133) makes a distinction between initial ordering and final optimization of weights ("fine-tuning"), the latter often taking considerably more time than the former. This costliness is magnified by the load of using matrix comparison with every element of the input sequence. Hence, some attention will be given to procedures that might help keep the computational load as small as possible.

Let us now turn to the notions of similarity measurement and pattern distances. The metric by which a SOM depicts the topology of input data should in some way reflect a metric similar to that involved in the cognitive processes of listening to music. The simplest case of referentiality, of course, is that of identical-feature vectors. Identity checking is trivial, and easily accomplished by even the most modest pattern-matching systems. The nontrivial part – recognition of different variations and degrees of partial correlation – reveals the strength of the system. In the finer shades of recognition lies the justification for such a compositionally laborious procedure as SOMs.

Its pragmatic nature is one way in which the present SOMAT implementation deviates from the standard SOM. An important point, which Hecht-Nielsen (1991: 68-69) has noted, is the possibility of overly large differences between the distribution of feature vectors, in comparison to the distribution of weight vectors as they are originally initialized to random values. Such situations may lead to poor representations of the pattern space. He reports the solution given by Duane Desieno, which is a "conscience mechanism" that prevents each unit of the layer from too often overtaking the others in excitatory response. In the present context, however, such finesse may not be necessary. The purpose of SOMAT is to make use of the adaptation and unsupervised-learning capabilities of the Kohonen layer, while not attempting to construct a set of equally probable, distributed weight vectors. The computational load thus becomes substantially lighter.

The main technical differences between SOMAT and standard SOM implementations are the following. Kohonen (1989: 127-133) uses decreasing learning speed as well as shrinking neighbourhood radius for the lateral interaction of units. This is to ensure that the system converges to a particular topological ordering instead of drifting aimlessly from one ordered state to another; it also enables the fine-tuning of weights, while retaining the main contours arrived at in teaching. In the designing of SOMAT, it was felt that finely tuned ideal topological order was not of primary importance. Thus, variable neighbourhoods, as well as decrease of learning speed, were done away with. Another mechanism was devised to compensate for the shortcomings of ordering. I call this mechanism *shadow activation*.

In the experiments, it was observed that the mismatch between weight-vector distribution and pattern feature-vector distribution was dependent on the lateral feedback function chosen for the system. There are several workable ways of implementing lateral feedback in the layer. Even in Kohonen's original account of the SOM he feels no need to specify lateral functions beyond general guidelines (1989: 127). In many cases, it seems likely that the lateral function can be chosen so that the distribution mismatch will not be large enough to distort the behaviour of the system significantly, at least from the point of view of shadow activation.

Now it is time to define the way to measure pattern distances by means of the Kohonen layer, while using minimal effort in teaching cycles. The SOM seems to lend itself well to pattern matching via the topology of the layer – the distance between excitatory foci for two feature vectors acting as the measure of pattern distance. In this experiment, however, the objective of ideal ordering of adaptive units was set aside in the pursuit of computational efficiency. The layer was used with significantly less training. Furthermore, in layers with a relatively small small number of elements, only a coarse classification is possible on the basis of Euclidean (or some other metric) distances on the layer. Therefore the pattern distance has been defined differently:

**Definition 7.1:** *Shadow activation* of feature vector $F_a$ in respect to feature vector $F_b$ is the amount of activation invoked in the element specialized in responding to $F_b$, when a trained self-organizing layer is stimulated with $F_a$.

Shadow activation, as manifested in the present implementation of SOM, can be thought to have a biological interpretation, too. It closely resembles the so-called *afterimage*, the neural phenomenon wherein a visual image remains after the stimulus that caused it has disappeared.

Shadow activation can provide a measure for pattern distance. It performs in ordered fashion and is distinguishable after much fewer training cycles than are required for an ideal topological organization of the layer. Yet it retains some of the distributed information gathered in the adaptation process, which takes places during the teaching of the map, thus exceeding the flexibility of any conceivable non-adaptive process.

## 7.2 On structuralist ideas, holism, and transience

Throughout the twentieth century and continuing today, an unforeseen process of specialization has had a pronounced effect on most branches of science and the humanities. At times, this analytical spirit seems to split and shatter every effort to perceive the world as a whole. There has been a counterreaction to this analytical tendency, however, and parallel distributed computing is but one representative of an ongoing revival of *holistic* ideas in scientific thought. The desire to see natural phenomena as totalities can be traced back to antiquity, but during the last two or three centuries has somewhat been eclipsed by the desire of mechanistic minds to analyze the world by dissecting it. Among other twentieth-century intellectual movements, structuralism stepped forward to emphasize the importance of looking at various phenomena as totalities. Michael Lane portrays the objects of structuralist study as "expressed wholly in the relations constituted between them" (1970: 24). Certain structuralist ideas curiously foreshadowed technological advances in the 1980s and 90s in connectionist computing and parallel systems, which share some of structuralism's non-historical – some might even say, non-causal – nature.

Structuralist ideas have left their mark on music studies, often in connection with the semiotic research of scholars such as Jean-Jacques Nattiez, David Lidov, Fred Lerdahl, and Ray Jackendoff, to mention only a few. In the spirit of musical semiotics, we often speak of two dimensions when inquiring about musical sequences: paradigmatic and syntagmatic. In the case of "intelligent machine-driven" segmentation, the pursuit of syntagmatic (sequential) relations is the initial step of the procedure. Only on the basis of syntagmatic results, derived from segmentation, can paradigmatic relations be defined.

The subject matter of syntagmatic analysis is segmentation, i.e., locating the boundaries of segments. Because we wished to account for multiple variants and similarities among segments, it was decided to use an SOM in order to have a flexible measure of pattern-similarity. By traversing the syntagmatic line, from the beginning to the end of the time sequence, we detect segment boundaries, or transition points from one segment to another. To be able to detect them, we needed a way to measure the "transitoriness" of elements (time positions) of the temporal sequences. Let this quantity be called *transience.* It is understood as a scalable quantity related to each particular time position of the sequence.

Transience is not formalized to a precise measurement here. Calculating absolute transience values is not possible without further specification, and it is not certain that attempting such a calculation would serve any purpose. Here we simply introduce a means of approximating such a quantity in a relative way, which is sufficiently formal for algorithmic treatment.

The strategy is as follows. The algorithm described in the previous section (7.1) outputs a resulting signal, which is a discrete function representing the changes of transience, computed from the input signal. The transience value at any point $n$, corresponding to the element $S[n]$, is related to the number and length of correlation chains pointing at $S[n]$. These values constitute a new signal reflecting the transience properties of the original time sequence, which will peak at the segment transitions, thus producing a syntagmatic division of the original message. Every element of the sequence is computed against the context of all other elements; hence the ground for forming the transience signal is effectively all of the original signal. The aim is to propel the procedure by means of holistic, parallel detection of relations present in the input signal. Figure 7.2 visualizes the division of the analysis machinery into subsystems.

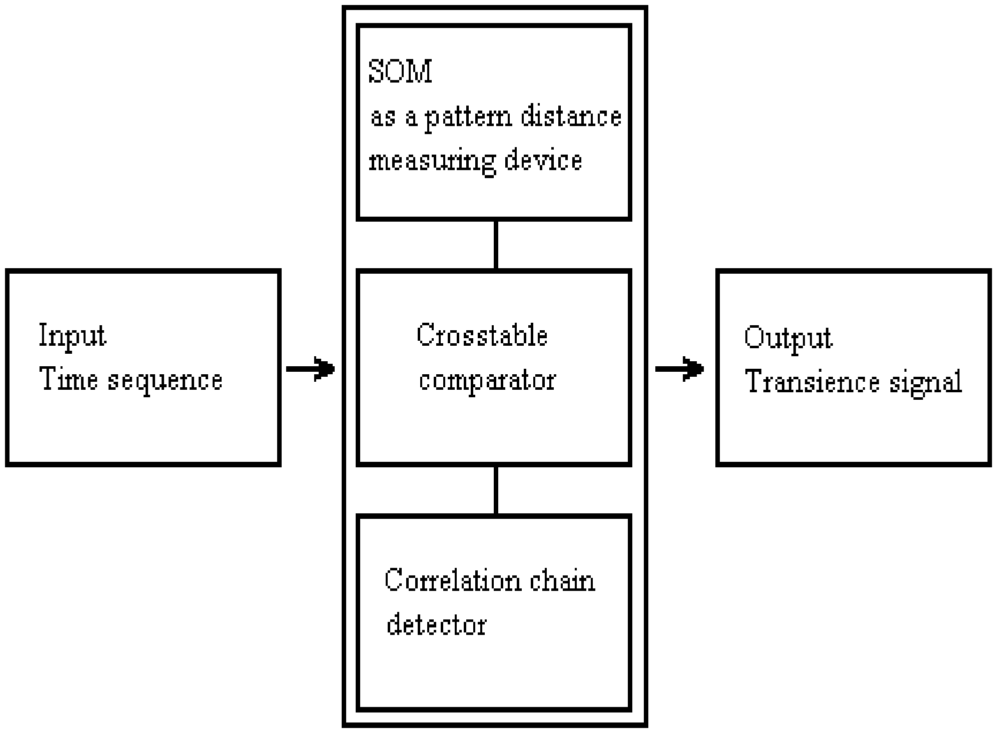


**Figure 7.2:** Time sequence analysis using SOM: Diagram of SOMAT subsystems.

Tenney and Polansky (1987: 216) introduce a quantity called *disjunction*, which is used in a way that is comparable to our notion of transience. The authors seek to give a specific, quantitative definition in terms different from the ones adapted here, by using Gestalt principles. Therefore, it is not feasible to use their terminology at this time, though the role of disjunction in their theoretical construct is very much related to transience. Tenney and Polansky's term *segregation* is also used in a rather similar way. Also, Cambouropoulos (1996a: 282) speaks about *boundary strength-values* in a sense comparable to that of transience and disjunction.

## 7.3 Experiment reports: SOMAT at work

### 7.3.1 Principles and parameters

In these experiments, the input sequence is taken from the pitch information of a musical score. Absolute pitch is used, with no information concerning octave equivalency or related notions, such as pitch-class or key. Because the purpose was to feed truly monodic material into the system, the example material was chosen from the solo flute repertoire. This material was considered suitable not only because of its lack of polyphony (and multiphonics as well), but also because of less variation in the treatment of timbre than is the case with many other orchestral instruments. The examples were chosen from twentieth-century music, because it is conceived as being less regular in structure, and thus more challenging as concerns formal aspects, than is music of, say, the classical period.

The SOMAT program allows for several parameters to be changed. Some of these changes are only to facilitate its practical usage, while others have a pronounced effect on segmentation. The substantially important parameters also greatly influence the running time, so there are practical limitations to the values. These parameters are as follows:

-Number of training cycles for the SOM
-Number of components for the feature vectors (equivalent to the dimensionality of feature vector space)
-Size of the SOM (length of one side of a square)
-Coefficients determining the shape of the lateral function

The lateral function creates an environment of 3 concentric circles around a central node. Thus the furthest nodes of the environment are located 3 nodes away from the center. Larger environments would actually be nonsensical, given the relatively small map sizes used in the experiments.

The series of tests reported here used mainly a map size of 18, for a total of 324 (18 * 18) nodes. Only 30 teaching cycles were used, to provide a very rudimentary organization of the map. The system was run with two sets of parameters, each set with a different setting for dimensionality. Higher dimensionality yields higher convergence to a specific state of the system, whereas lower dimensionality means lesser specialization and higher generality. The first of the two sets of parameters uses lower dimensionality; the second set uses higher dimensionality. This is the sole difference in the runs. Other user-set parameters of the program include size (number of nodes) of SOM, coefficients of the lateral feedback function, and number of training cycles in the teaching mode.

### 7.3.2 Paul Hindemith's *Acht Stücke*

As another example, we present Number 6 of Paul Hindemith's *Acht Stücke*. Before going into the SOMAT process, we make some observations concerning the context. The piece belongs to a cycle of eight miniature pieces for unaccompanied flute. It is quite brief, its length spanning only 14 bars. In all eight of these small pieces Hindemith builds form and coherence by classical means of repetition and variation. Nevertheless, the models and templates used for harmony, melodic movement, and treatment of rhythm and form in the piece are not reproductions of classical models. Instead, and despite of the likeness, more degrees of freedom are allowed than in the classical vocabulary – exactly what one would expect from someone committed to neo-classical means of expression. There are gradually-moving, chromatic sequential patterns, which deviate from traditional tonality. The composer uses gestural or vector-based[17] movement as a tool for building form. Anaphoric repetition is also an important compositional device in some of these pieces. These methods of repetition and variation are used differently in different parts of *Acht Stücke*.

Let us now briefly consider the character of each piece in the cycle. The piece no. 1 employs sequential patterns repeating at different pitch levels. Repetitions of the same phrase appear 2-3 times. No. 2 is similar to Number 1,

[17] The term *vector* is not used at this instance in the strict mathematical or formal sense. It is used to mean a general tendency of movement from one pitch level to another; typically from the apex to the nadir of a pitch pattern, or vice versa.

in that the same phrase appears two or three times. No. 3 has a characteristic interval, the minor second. Its chromatic nature might make an astute listener wonder if a twelve-tone row is involved; but in closer study reveals that this is not the case. A salient theme and variational tendency exist, and the structure is built largely on variation. Piece No. 4 is built on triple repetition of a single phrase, the repetition being somewhat anaphoric. The number 3 seems to be the important notion here, as far as form is concerned. Piece No. 5 contains double, triple, and quadruple repetition, such that a final repetition is linked to the first appearance of the next repeating pattern. No. 6 is the most conventional of the cycle, in its treatment of rhythmic structures. The time values could be a direct quotation from Mozart or Haydn. Only the chromaticism identifies this melody as one of the twentieth century. Piece No. 7 appears to be the most capricious one of the set, due to its numerous variations in time values. Here the repetition is extended to such small temporal units that it almost ceases to be a form-building element. Piece No. 8, the finale of the set, brings together materials that have appeared in the previous ones. The contrast of among time values is not as great as in the previous piece, yet a similar feeling of caprice is created by sudden accents, dynamic change, and free treatment of rhythm. The cycle ends with two extensive accelerandos and a presto cadenza.

The material of No. 6 of the cycle is derived from two simple motifs which appear in the first two bars (see Fig. 7.3). Both of the basic motifs fit the 4/4 time signature comfortably. Both are primarily rhythmic in character, and undergo major changes in pitch structure during the piece. Thus, despite its seeming simplicity, this example may present a considerable challenge for a segmentation system based primarily on pitch information.

Before going into the results of the computer runs, let us first study the melody “manually”. One way to interpret the material would be as follows. Motif *a*, which begins the first bar, consists of one crotchet, a dotted quaver followed by demisemiquavers, ending on another crotchet and a pause. The general direction of pitch is descending. Motif *b* consists of two quavers, followed by four semiquavers, followed by a quaver. In this motif, too, there is a tendency to descend in pitch.

After introducing the motifs, the melody expands its pitch range with an upward scale in m. 3, which brings the melody to a variation of motif *b* in m. 4. The scale itself is quite diatonic, in a slightly unexpected way, with a mixolydian flavor. Measures 5 and 6 reintroduce motifs *a* and *b* with a slight variation.  Measures 7 and 8 introduce contrasting material, which, however, is easily related to both of the main motifs.  In mm. 9 and 10, motifs *a* and *b* are repeated in their original form, leading to a development in mm. 11 and 12. The piece ends in mm. 13-14, with material that seems to be derived from motif *b*.  Thus, the form may be understood in conventional terms, as built on phrases of 2 bars each.  Yet because of the development of material, not all of the two-bar divisions are equally important.  The recurrence of the original material at the beginning of m. 9 is likely to be heard as weightier than others, such that the total length of 14 measures can be divided into the proportion 8:6, which is the major division of time.  A deviation from the normal two-bar units can also be distinguished on the last beat of m. 11, which stands out as the beginning of the recapitulation of the closing phrase (which is actually derived from motif *b*).  Thus, the phrase starting at the beginning of m. 9 would also include the first 3 beats of m. 11.  The ending, from the last of m. 11 through m. 14, is also likely to be heard as one phrase.

Of course, the foregoing analysis is not the only possible interpretation of the piece – others may well be derived on the basis of the score (Fig. 7.3).

The present computer simulation represents activation as integer numbers ranging from 0 to 99. The choice of integers was based on computational efficiency. The upper limit of 99 is rather arbitrary, but the overall range is believed to be a safe one in which this particular system will work, without the undesirable side-effects of an overly coarse weight-adjustment taking over the process and causing instability.

The map itself is the 18 x 18 matrix of 324 nodes total, as was already mentioned. In order to avoid undesirable effects caused by incomplete environments close to the edges of the map, the latter effectively forms a torus shape, so that left and right edges are adjacent, and the top and bottom edges as well. In the model, each feature-vector component represents the duration of 1/32-note (demisemiquaver). Since the time signature of the piece is 4/4, each bar is represented by 32 feature-vector components, whereas the number of feature vectors in the bar depends on the choice of vector-space dimensionality. The vectors used for the example consisted of either 3 or 6 components.

Two outputs of the SOMAT system for No. 6 of *Acht Stücke* are shown in Figure 7.4. The figure consists of three curves. The top one represents the input signal, which is the extracted pitch information. The second curve represents SOMAT output with 3-component feature vectors (3-dimensional space). The bottom curve represents SOMAT output with 6-component feature vectors. The expectation is that the latter output signal should give more precise information, since the specialization of 6-component vectors is greater than that of 3-component ones. Some smoothing (made with simple 1st order low-pass filtering) and scaling were used in creating the curves in order to make them easier to read. Otherwise, they are faithful reproductions of the signals.[18] The horizontal positions of the curves are locked to each other, so that approximate comparisons may be done on the basis of the figure.

[18] One aspect of making such graphical images is that, if nonlinearities such as signal-clipping distortion creep in to the process, they may be hard to see from the images, but may still greatly affect the conclusions drawn from the images. As a careful reader may notice, some clipping distortion is present in Figure 7.7 (Saariaho’s *M. C. Escher*), but is left in place because of the rather large dynamic changes between the beginning and ending of the piece.

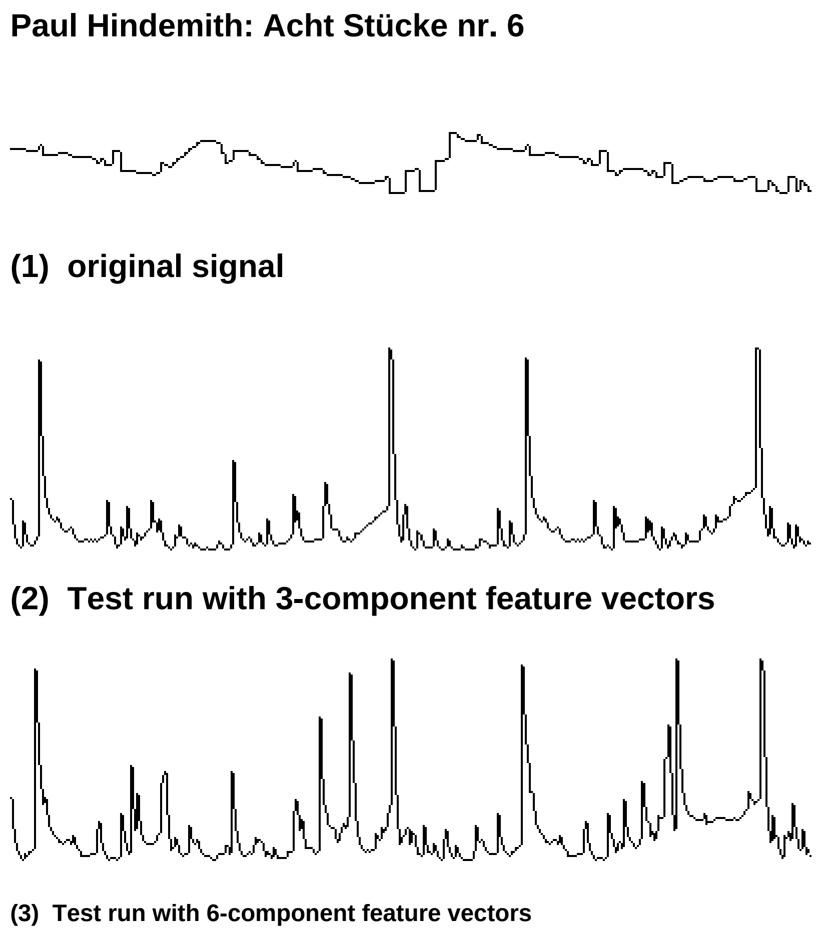


**Figure 7.4:** SOMAT input and output signals in graphic form.

Figure 7.4 does not include any means of facilitating interpretation of the curves, such as musical notation or bar lines. Nevertheless, because of the vertical coincidence of the curves, and with careful examination of the figure, approximate conclusions may be drawn. Segmentation between principal motifs *a* and *b* is recognized by both the 3-dimensional and the 6-dimensional cases. Moreover, the latter case also clearly recognizes the end of motif *b*, while the former is not as clear in this respect. Both recognize the transposition of these motifs in mm. 5 and 6, but locate the beginning of a segment slightly early. The 6-dimensional case again notices the change from motif *a* to motif *b* at the beginning of m. 6, and both recognize the change to developmental material at the beginning of m. 7. The large division of the melody takes place at the beginning of m. 9, which repeats the beginning motif.

Both the 6-d and the 3-d case recognize this division, but this cannot be taken as a significant achievement for a learning system, since this is an identity check and could be recognized by just about any existing pattern-matcher. The slight change between mm. 10 and 11 is recognized by 3-d analysis, whereas the 6-d case misses this change, and prefers to set the segment border to the last crotchet of m. 11; this might in fact be a good choice, for reasons already given in the “manual” analysis. Both cases recognize the last two bars as forming a single segment.

```
236936   11588   10546    7988   12104    2621    3392   23030
  9421   27625   15274   23861  702910  167912  106501   95951
 75767  134509   65053   52439   56652   46066   45232   39458
 33974   42552   36074   44136   49951   44992   22785   67012
 18763   20962   16073    6069   14137    8193   14490   11759
 20140   15892   21325  105689   95693    3529   12893    8910
  5835    5123   11789    7639   10589    2536   14217  174056
 31940   19216   12562   13234  345654   28218   21024  202341
 37477   24747   30438   30630   30921   32004   34173   36831
 40154   64862   57528  245894  173803  176167    8058    3531
  4554   34627   60706   28077    4317    7935    7549   18082
 15998  125165   26673   30886   37320   46177   13324   12131
 10253    8669    7771    8444    6877    6877    3168   20463
 15026   20641   20430   48044   31365    5111  329584   35213
 19667   21745    7677   14970    9668   10539   16301   26364
  9019   45033   63680   35333   51477   35747   10832    5917
 23291   12033    4122   45631    5679    6375    8060    7587
  9640    9493   32038   20225   15437  167103  140967   67150
 35369  101526   13524    9084   26054   25137   24837   13929
 13159   27530  520133   46592   61908   81483   46527   64758
 65643   67122   23856   37362   73176   76359   77533   54608
 98878  627022  110090   70666    8790   22176   22374   21186
 20493   19995   23519   21843   23088   25270   87636   39368
 42860   84991   45108   73251  116527  124995  907252    8688
  7092    5322    6234   56174   54364   69951   74405    2829
 81212   45173    9077    5977   15510   11185  124037     594
  1782   20513    9946   57450   13055    1760    7454    1848
 76022   77996    5088   13014     880    4321   53085   14844
  5128    4534    4022    1964    1172    7035    3316   33917
116655   21507   32120   51833   22321   20830   19719   13231
 19460   19024   16338  165781   14914   15964   10210   16737
  8516    9875   29814   16995   35516   23359   32068  711146
191333  167688  157429   77047  134509   65053   52439   56652
 46066   45232   39458   33974   42552   36074   44136   49951
 44992   22785   67012   18763   20962   16073    6069   14137
  8193   14490   11759   20140   15892   21325  105689   95693
  3529   12893    8910    5835    5123   11789    7639   10589
  2536   14217  176119   28467   36002   15981   13248   45018
 41881   18394  209856   20133   30089   32879   32673   27109
 29301   30409   53227  261833   66359   54803   71914   29134
 66746   28850   68452  114150   78370   84595  303938  199175
317515   87027    8698   31856  909363   32271   91572   88899
 87810   85236   86817   85729   83946   79692   78699   78496
 79782   78976  104527   54150   77712   79296   80583   89889
 89683   84540   85432   86718   83652   82461   82159   81666
 81850   84577   76824   80979   84341   91356  101537  100883
167017  107143  115311   99271  115428  120883 1286007   17076
 44992   14929   18398   15804  138868    5772   54196    4683
  2667    4146   92402   37765   71319   28617  179379   27956
 24343   14929   17606   98265    6084    2541   32266    4683
  2667    4146  108590   57109   83751   20244  187849   30405
 24343   14929   19483  100938    9442    2541   45532    6294
  6906   16113   17499   19578   22452   29757   59286   29076
 23256   22566   22569   23265   24651   26136   31179  128892
```

**Figure 7.5:** Output signal of 6-component vector-process shown in numerical form

Should the reader be disturbed about the effect of the smoothing process used in realization of the curves, the raw numerical data for one of the curves are presented in Figure 7.5. These data are given in 8 columns, each number representing a time value of 1/32-note. Each row represents a quarter-note time value, and each of the 14 bars of the melody, in 4/4 meter, occupies 4 rows.

## 7.3.3 Claude Debussy's *Syrinx*

Another test run used input data from Claude Debussy's well known solo flute piece *Syrinx*. The results are given in Figure 7.6. The SOM used is a 324-node 18*18 layer, as was used in the Hindemith example. The meanings of the three curves in the diagram are also similar to those in the Hindemith piece. The top curve represents the contour of the melody (input signal). The other two curves display the two output signals, resulting in the two control-parameter sets. The middle one represents the results with 3-dimensional feature vectors, and the bottom one displays the results using 6-dimensional vector space. The directions for reading the curves are similar to those of piece No. 6 from *Acht Stücke*.

Though neither of the runs produced a perfect analysis, both outputs bear more than incidental resemblance to the phrase structure of the melody. The output produced by 6-dimensional feature vectors seems more musically justifiable than that of the less specialized, 3-dimensional representation. A number of reasonable divisions resulted, along with some clumsy ones. Clearly differentiated are the first phrase, which introduces the main melodic ideas of the piece (Austin 1966: 7-13), its repetition, and its development. The next division is not in the correct location, since it does not recognize the unity of mm. 1-8 and their difference from the variant of the first idea as developed in mm. 9-14. Then, a short bridge which follows (in mm. 14-15) and the contrasting middle section of the piece (mm. 16-26) receive fairly clear recognition, as does the melodic climax at m. 27, the very significant moment described by one music scholar as "stretched out to extreme poignance" (ibid.). In the last part of the melody (mm. 28-35), which project an affective sense of resignation, the program easily discerns the short, individual phrases.

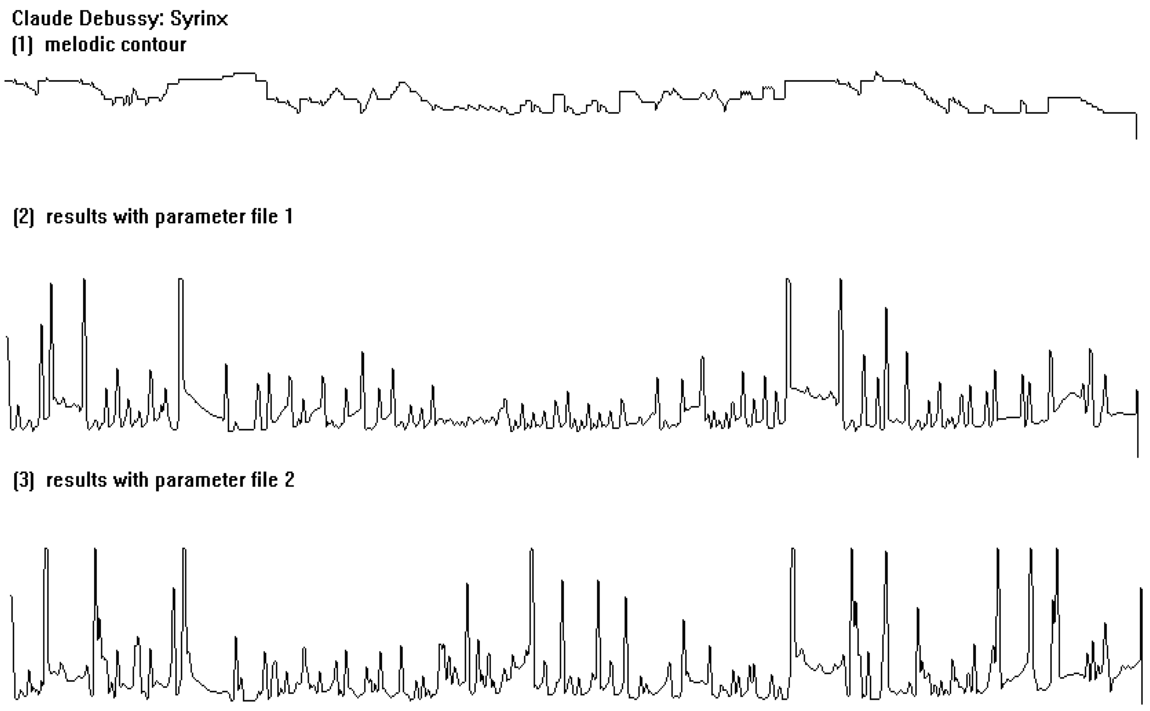


**Figure 7.6:** Test run on Debussy's *Syrinx*

In comparison to the results obtained by Tenney and Polansky (1987: 232-234), it may be said that, as concerns the most obvious segment boundaries, their system seems to have performed more reliably than our own. They present a more hierarchy-oriented segmentation, whereas the output of the present project does not make much distinction between different levels of form. However, SOM-based pattern recognition seems better than Tenney and Polansky's model at recognizing the less clear boundaries. Yet it is hard to declare one system to be "better" than the other. This is because optimal segmentation often is not very clear on the basis of music itself, and ambiguity is an integral part of certain musical *paroles*. In any case, a considerable amount of test material would be necessary in order to make a definitive judgement. On the basis of this experiment, however, it appears that the SOM-based system might serve well as an adaptive, sequence-segmentation machine.

## 7.3.4 Kaija Saariaho’s *Canvas*

The next piece of music is rather different from that of Hindemith and Debussy. Kaija Saariaho’s *Canvas* is a set of three pieces for solo flute, which demonstrate composer’s fascination with formalism and abstraction. The first piece of the set, subtitled *M. C. Escher*, will be studied here. Again, only pitch information (and, indirectly, timing information via time-slicing) is considered in building the input signal for SOMAT. This is definitely a loss in terms of interesting information being reduced away, since the piece makes heavy use of ties, staccatos, accents, and dynamic markings. No traces of diatonic scales are distinguishable. Permutation is used extensively, as are contrapuntal variation techniques. Octave equivalence, in the serialist sense, is used abundantly. The piece is built around a single motif, which in its basic form consists of four 1/8-notes E flat, C, B, and E. It must be noted that in many European languages including Finnish, E flat is read as Es, and B as H. Thus the motif takes the form *Es-C H-E,* derived directly from the name *Escher* of the subtitle. Any meanings of this kind are based on arbitrary (from the point of view of music listening) assignment of significance, and as such are purely symbolic. Therefore SOMAT, which is built to detect iconic information, has no chance whatsoever of understanding such meanings. Also, the abstract nature of the compositional tools used in this piece may present a challenge for the system. After all, being able to distinguish immediately the variation techniques treasured by serialist composers, such as permutation and inversion, is not self-evident to listeners, not always even to musically educated ones. SOMAT is essentially a “musically uneducated, time-sequence listener that has a simple memory mechanism and that is prepared to educate herself”. What exactly is left, then, for the computer to analyze, after leaving out the possibility of understanding notational cryptographs and puns? The time sequence will represent patterns in time, vectors, and tendencies just as it did in previous examples. Furthermore, matters such as the messages conveyed by these forms and their interplay are probably closer to the essence of music as an art form of than is the solving of verbal puzzles and puns. To make sense of such matters continues to be the objective of the system. The output signals and the input signal of *Canvas* are displayed in Figure 7.7.

The piece has no barlines or time signature. Lightly dotted lines are provided, in lieu of traditional barlines, in the version of the score that was used for input data coding.[19] Obviously they serve to help the instrumentalist practice the piece. In other editions, as a rule such a division always comes after 8 quavers, so this "virtual pulsation" largely follows a time signature of 4/4. Exceptions this metric pattern occur, however, in mm. 9-15 and 24-29, which introduce rhythmic augmentations of the material. Still, let us refer to these as "bars" or "measures", since they actually belong to the music. There are 33 such bars in Part 1 (*M. C. Escher*) of *Canvas*.

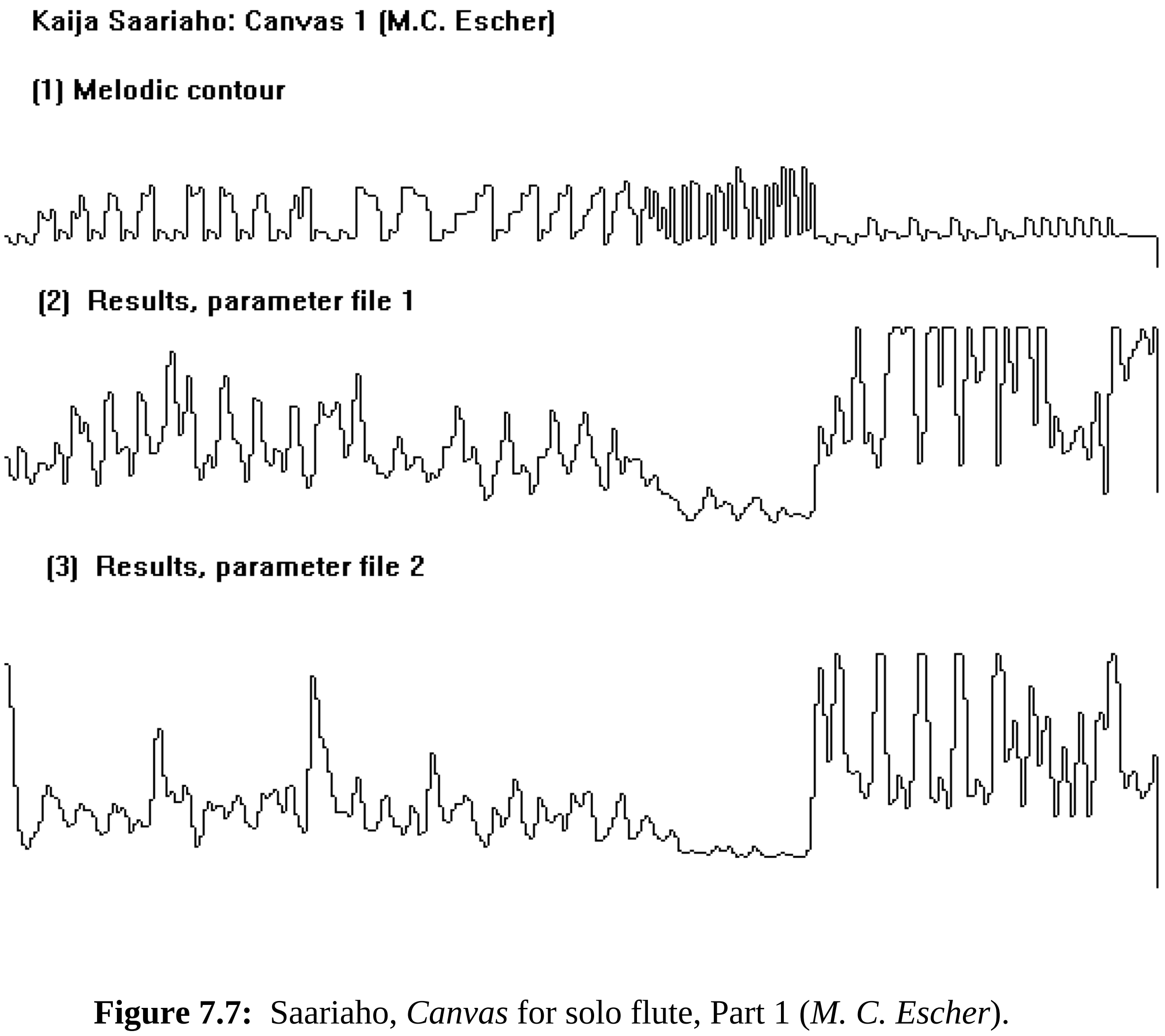


**Figure 7.7:** Saariaho, *Canvas* for solo flute, Part 1 (*M. C. Escher*).

[19] This was the version available at the Finnish Music Information Center.

The treatment of material has other connections to serialism, too. In mm. 1-2 the motif *Es-C H-E* is repeated twice, then once more in octave transposition. Then comes the first permutation, transposed up two semitones from the original pitch. The motif is such that the original and the transposition together fill eight successive positions of a 12-tone scale, a chromatic scale segment covering B to F sharp. The rest of the 12-tone chroma – pitch classes G, G sharp, A, and A sharp – do not appear at all in the piece.

The 3-component case of the SOMAT output signal clearly omits salient events that should be recognized. The distinction between higher and lower peaks is not very evident, and there are too many of them to make far-reaching conclusions. My visual inspection of the score reveals two major formal divisions. The first one is at the end of m. 9, which is the conclusion of a strong ascending motion. After this local high point in pitch, the melody returns to the original register at the beginning of m. 10. Another significant division of form occurs before the beginning of m. 24, at the *a tempo* marking, which, after the development in the middle section of the piece, returns to the basic form of the *Es-C-H-E* motif.

The 6-component case of SOMAT output signal seems more successful. The big segment boundary at the start of m, 10 is recognized perfectly, as is the beginning of m. 24. There is a problem, however: in the final part of the piece the program indicates other strong boundaries, which are not completely compatible with listening or studying the score. In addition, one slightly weaker boundary is indicated at the center of m. 5. The latter boundary, however, is possible from the perspective of listening, and perhaps should not be counted as an actual fault or misbehavior.

## 7.4 Discussion of test runs

As expected, in all three experiments the 6-component case of the SOMAT output signal performed better than the 3-component one did. In the Hindemith example, the difference between the 3-component case and the 6-

component case was not very large. In *Syrinx*, the 3-component cases lumps together the whole middle section of the work, from m. 9 to m. 24, as not having significant segment boundaries. This is obviously not right, and the 6-component case solves the problem. Part 1 of Saariaho's *Canvas* seems to be the hardest of the three music examples for the system to process. The results of the 3-component case are quite unclear, whereas the 6-component case produces a reasonably intelligible output signal. As a conclusion drawn from the preceding discussion, we can say that 3-dimensional space does not seem to be able to provide sufficient specialization for feature vectors.

A general assessment of the performance of SOMAT system is that it performs remarkably well, considering them narrow slot of information given to it. The fact that increasing the dimensionality of the vector space enhances the performance, as it theoretically should, suggests that the system is built on sound principles. An in-depth series of experiments, using larger maps and additional teaching cycles, would certainly reveal more about the functionality of SOMAT. In settings such as the ones described in this chapter, it probably cannot outperform rule-based systems. To do so, however, was not the point of the experiment. Originally, the objective was to collect empirical evidence for the hypothesis: that a system for time-series analysis, using non-supervised learning, and justifiable both in the light of Peircean semiotics and parallel computation might actually work. For this hypothesis, the examples from *Acht Stücke*, *Syrinx*, and *Canvas* give strong, though not completely unambiguous, positive support.

The example of *Canvas* also pointed up the inevitable limitations of the system. Purely symbolic information lies beyond the scope of SOMAT. Also, to what extent the system can handle different variation techniques, was not determined by these experiments. The best way to draw any conclusions about such matters would probably be to construct artificial sequence examples instead of using "real" music. In that case, it would be possible to isolate the effect of multiple influences on recognition, and to concentrate on one variation technique at a time. On the other hand, due to the holistic nature of the system, any performance in a real musical situation might differ from such artificially constructed tasks. Also, what makes this study belong to musicology, instead of computer science, is the very use of data originating from real music, and the choice of material that may be expected to challenge the performance of the method on musical grounds.

The test runs on the three melodies were conducted originally at the same NeXT workstations that served as development platforms for SOMAT. Thus, the computing power was severely limited, by present-day standards, and no possibility existed for examining the behavior of larger networks or higher-dimensional spaces. Additional runs were later made using a Silicon Graphics Origin 2000 supercomputer, which allowed those aspects to be studied. The input signal for the latter test runs was taken from Hindemith's *Acht Stücke*, No. 6. Figure 7.8 presents the resulting signal for a network of 2500 nodes.

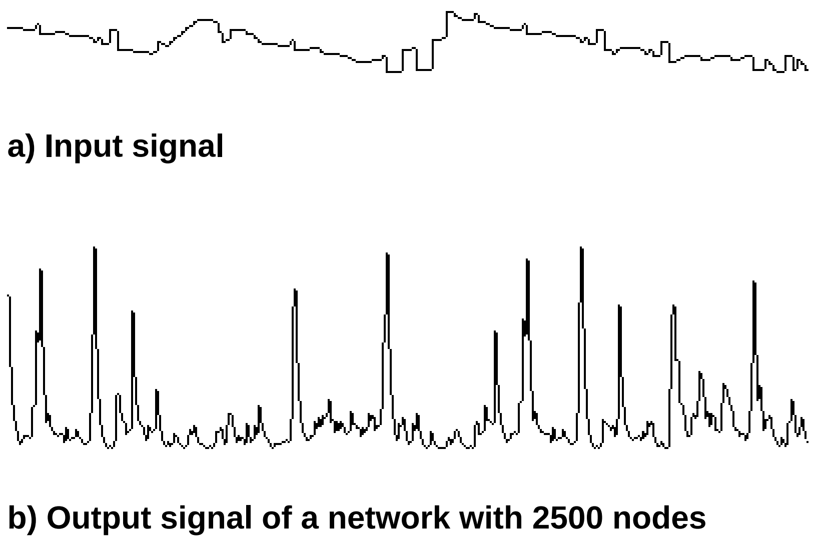
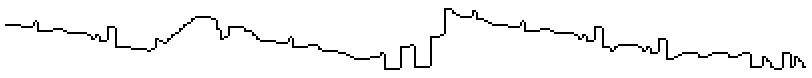


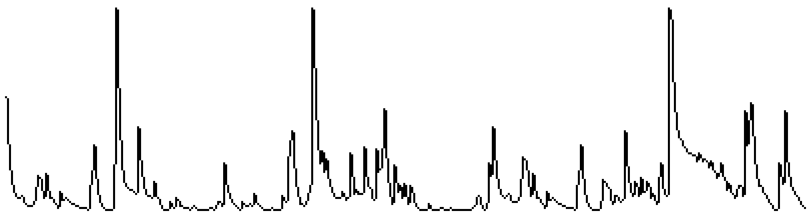
**Figure 7.8:** SOMAT input and output signals for *Acht Stücke*, No. 6, using a larger SOM.

A general observation about the behavior of the larger network is that it seems to produce more peaks with the small amount of training given. This accords with common sense about the matter: it is easy to suppose that a larger number of processing elements is apt to produce more noise and require more teaching cycles to achieve the state of organization equivalent to that of a smaller network.

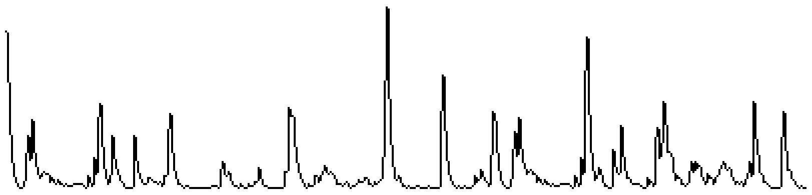
Still more test runs were made with more teaching cycles and higher space dimensionality. The output and input signals are given in Figure 7.9.



**a) Input signal**



**b) Output signal of a network with 2500 nodes and 300 teaching cycles**



**c) Output signal of a network with 2500 nodes, 300 teaching cycles and 10-dimensional vector space.**

**Figure 7.9:** SOMAT input and output signals for *Acht Stücke*, No. 6, using a larger SOM, more teaching, and higher dimensionality for the feature-vector space.

These last examples confirm the notion that putting more teaching cycles into the larger network will produce a more organized state. Also, they point up the fact that having a larger network does actually not do any good if all other parameters remain the same (including the input signal), unless more teaching cycles are used in training the system.

The output signal depicted in figure 7.9c also leads one to believe that increasing dimensionality (adding more components to the feature vector) enhances precision of the process, since the patterns represented are more specialized. To give more precise and certain information about the process, a more thorough testing program would be necessary. The test run of Figure 7.9c locates the major formal division at the beginning of m. 7. If this result were taken as an approximation that includes the last two semiquavers of the previous bar, then the solution would be musically credible, though it is not located as nearly to the Golden Section as is the return to motif *a* at the beginning of m. 9. The latter bar is where I am personally inclined to place a major division. In any case, the division at m. 7, or actually at the end of m. 6, is plausible from a musical point of view.

# 8 Technical implementation of the experiment: The SOMAT program

## 8.1 General

The program called SOMAT (Self-Organizing Map with Association-Tracing)[20] was created as part of this study and was used in the experiment described in Chapter 7. SOMAT was written in C programming language in UNIX environment. The test runs were carried out in NeXT machines, but the code follows standard C and is designed with cross-platform portability in mind. User interface is a plain command line one. Developing a graphical user-interface was not attempted, because of the long execution time of the software, and also in the interest of total portability. In the past, the demand for computing power has also ruled out the use of some low-end computer platforms, depending on the amount of input data. An MS-DOS compiling option is also provided as macros in the code, but because UNIX systems were available, which are generally more usable and manageable in this type of work, the SOMAT program has not been tested in MS-DOS.

The code was structured to different levels, following modular programming practice: the definition and implementation of a single node and an entire map, respectively, have both been separated from the main control

[20] SOMAT is essentially a revised and more developed version of the SONA program, introduced in the present author's earlier work (Tiits 1994).

structure of the program. Both node and map occupy their own implementation and definition files – `node.c` and `node.h` for node, and `map.c` and `map.h` for map. The main program and main control structure are in a “workbench” file called `wbench.c`. There is also a general header file `somat.h` for global variables and a definition of the data structure for holding them. Documentation of the software is given in this chapter, and the C language source-code can be found in Appendix 1.

The general logical structure of the program is divided into three sections: 1) input and output; 2) initializing and teaching the map; and 3) using the map to measure pattern distances and to produce analyses of connectedness between different subparts of given streams of input data. In Chapters 4 and 6 it was explained that, to feed input to the map, the data are broken into *feature vectors,* the latter being short data vectors of constant length, which provide a sliding time-window on the data. The number of components in these vectors corresponds to the dimensionality of the vector space used in computation.

When running, the program creates a 2-dimensional Kohonen layer with randomized weights, reads input signal, adjusts the weights, measures shadow-activation values for independent feature vectors via a matrix cross-table comparison procedure, and writes the output signal. The running time may be considerable, depending on the length of the input sequence, number of teaching cycles, and size of the SOM.

## 8.2 Input and output data

The program command line has the following syntax:

*somat parameterFileName < filein > fileout*

The parameters for SOMAT processing are given in a separate parameter text file instead of in a command line. The parameter file consists of lines, each containing a parameter name and its accompanying value separated by space. The name for the parameter file is passed as an argument to the command. Input and output signals are read from and written to text files at standard i/o. Thus, it is easy to use piping with the *somat* command and include it in shell scripts, should one wish to do so.

The following are the parameters and their meanings in the parameter text file:

| | |
|---|---|
| seed | weightRange |
| trainings | lateralFunctionGains (3 values) |
| spaceDimension | verbose |
| mapSize | bidirectionalProcess |
| outputFormat | |

*seed*

This parameter defines the seed for the random-number generator, which is used at the initialization of the weight-vector components of the map. The generated numbers are pseudo-random, so identical parameters will always produce identical network topology and identical results.

*trainings*

This parameter defines the number of training cycles. The number is used for changing the weights of the map according to the input signal.

*spaceDimension*

This parameter defines the dimensionality of the vector space used in describing the features, i.e., the number of components in the feature vectors.

*mapSize*

This parameter defines the size of the SOM used as the length of the edge of the map. A value of 10, for instance, instructs the system to build a square map of 10*10 nodes, or 100 nodes altogether. SOMAT always builds the SOM in the form of a square, the edges connected to their respective, opposite sides, so as to form a toroidal surface.

*outputFormat*

This parameter defines the number of columns used in formatting the output text file.

*weightRange*

This parameter defines the range of integer values possible for the weight vector components. The larger the value, the finer-grained and slower the adjustment process is. On the other hand, smaller values cause learning to be faster, but any non-linearities and possible artifacts tend to get stronger.

*lateralFunctionGains (3 values)*

These three parameters should be small positive or negative integer values. They define the gains for weight-vector changes that take place during the adjustment of weights according to the states of nodes in the neighborhood. Essentially, they define the shape of the lateral function. The neighborhood is defined as a three-stage one, with increasing radii around the center node. The first one of the gains controls the inner circle, the second one the intermediate circle, and the third one the outer circle of the neighborhood.

*verbose*

This parameter is used mainly for debugging purposes. A non-zero value will cause information to be printed out concerning changes made to weight vectors during the teaching process.

*bidirectionalProcess*

If this parameter is set to non-zero, the process will run from start to end, and then again from end to start.

Values of less than 6 for parameter *mapSize* are not allowed because of the three-stage neighborhood definitions in the lateral feedback functions. If a smaller value is entered, it is replaced by 6 to prevent neighborhood zones from wrapping around the torus. Very large values (30 or more) are likely to lead to combinatorial explosion, but the practical upper limit naturally depends on the performance of the computer system and the number of training cycles used.

The input file is a text file containing a stream of values from a time-series, such as the time-slices of melodic pitch shown in the example in Chapter 4. The output file is a text file, and in the present version of the program it first prints the reactions of the network organized as matrices of node states, one for each feature vector. Then the results of the analysis (transience signal values) are printed.

## 8.3 Structuring and the program flow

### 8.3.1 The contents of the source-code files and the function-call structure of the program

SOMAT functions are found in source files as follows. The function descriptions, set out in Chapter 8.3.2, are organized so as to correspond to this division.

Section 1: file `somat.h`: definition of run-time data structure for global variables of the system.

Section 2: file `node.h`: definitions of run-time data structures for an individual node of the self-organizing map.

Section 3: file `node.c`: functions that implement the properties and functionality of an individual node of the map.

Section 4: file `map.h`: definitions of run-time data structures for an entire map.

<u>Section 5</u>: file `map.c`: functions that implement the properties and functionality of the entire map.

<u>Section 6:</u> file `wbench.c`: a workbench with main program.for testing the functionality, and for operating the test runs.

```
  1     main: int (), <wbench.c 273>
  4         getParameters: int (), <wbench.c 213>
 10         quickRandom: int (), <wbench.c 13>
 11         initMap: struct nd *(()), <map.c 4>
 12             initNode: struct nd *(()), <node.c 5>
 14                 quickRandom: 10
 15         initDataVector: int (), <wbench.c 255>
 16         readVector: int (), <wbench.c 25>
 18         scaleDataVector: int (), <wbench.c 187>
 19             getMinData: int (), <wbench.c 167>
 20             getMaxData: int (), <wbench.c 177>
 21         trainMap: int (), <map.c 258>
 22             trainingCycle: int (), <map.c 240>
 23                 setMapResponse: int (), <map.c 66>
 24                     setNodeResponse: int (), <node.c 18>
 25                         abs: int (), <node.c 26>
 26                 getFocusNode: struct nd *(()), <map.c 165>
 27                     getPeakState: int (), <map.c 77>
 28                     getPeakXitationIndex: int (), <map.c 141>
 29                         getXitationIndex: int (), <map.c 101>
 31                         displayMap: int (), <wbench.c 43>
 34                     getXitationIndex: 29
 35                 teachFeature: int (), <map.c 218>
 36                     adjustZoneWeights: int (), <map.c 185>
 37                        adjustNodeWeights: int (), <node.c 30>
 38                             quickRandom: 10
 40         focusMap: int (), <wbench.c 197>
 41             setMapResponse: 23
 42             getFocusNode: 26
 43         classifyDataVector: int (), <wbench.c 150>
 44             searchReference: int (), <wbench.c 122>
 45                 getVectorCorrelation: int (), <wbench.c 94>
 46                     setMapResponse: 23
 47                     normalizeMapState: int (), <map.c 51>
 48                         getPeakState: 27
 49                         getBaseState: int (), <map.c 89>
 50                 searchReference: 44
 51         displayReaction: int (), <wbench.c 76>
 53             setMapResponse: 23
 54             displayPeakNode: int (), <wbench.c 56>
 55                 getFocusNode: 26
 57             normalizeMapState: 47
 58             displayMap: 31
 59         reverseDataVector: int (), <wbench.c 262>
 60         displayData: int (), <wbench.c 33>
```

**Diagram 8.1:** Program flow-diagram for the SOMAT system.

The function call structure of the program was generated with a UNIX utility program *cflow*, as shown in Diagram 8.1. In the diagram, all calls for C library functions have been omitted in order to enhance readability, hence the gaps in the ordering of line numbers.

### 8.3.2 Function descriptions

As a convention, large capital letters indicate types (e.g., `Node`), and small letters denote variables (e.g., `aNode`). Large capitals are also used to separate words into function and variable names (e.g., `weightVector`, `initNode`).

Section 1: file `somat.h`

*Description: File somat.h defines the run-time data structure for global variables of the system.*

All global variables are stored in a single structure `ParamStruct`. It is defined as follows:

```
typedef struct {
    Node *focusVector[MAXDATA];
    int seed, trainings, spaceDimension, mapSize;
    int outputFormat, weightsChanged, errorCount;
    int weightRange, lateralGain[3];
    int verbose, bidirectionalProcess;
} ParamStruct;
```

Section 2: file `node.h`

*Description: File node.h defines the run-time data structure for a node of the map. In addition, some constants and two global variables for the system are defined.*

The data structure for a node of the map is called `Node` by type definition:

```
typedef struct nd {
    int weightVector[MAXDIMENSION];
    int state;
    struct nd *n, *w, *s, *e;
} Node;
```

The vector `weightVector` contains input weights that correspond to each input of the node. Unlike many other implementations of self-organizing maps, the weights in this system are integers. The number of inputs is same as the number of components of the input data-feature vectors. Variable state stores the computed state of the node. Links *n, w, s* and *e* are abbreviations of the four main points of the compass – north, west, south, and east – and serve as links to the neighboring nodes. The immediate neighborhood of a node consists of eight adjacent nodes, four of which connect to the center node with pointer links; the other four links – north-west, north-east, south-west and south-east – are implemented as logical operations in file `map.c`. The constant, `MAXDIMENSION`, contains the maximum number of feature and weight vector components available.

Section 3: file `node.c`

*Description: File node.c contains the routines for initializing and teaching a node of the map:*

*Node *initNode(void)*
*void setNodeResponse(aNode,featureVector)*
*void adjustNodeWeights(aNode,featureVector,gain)*

*Node *initNode(mapSize)* allocates storage for a node of the map. The size of the map is given in the parameter `mapSize` as the length of one edge of the map when perceived as a flat rectangular surface instead of as a torus.

The routine does the following: it initializes pointers *n*, *w*, *s*, and *e* to NULL and state to 0; initializes `weightVector` components to random numbers between 0 and global parameter `weightRange`; returns a pointer to the node; is called by: `initMap()`.

*void setNodeResponse(aNode,featureVector)* computes the response of a node to a given feature vector and stores the value to the state-variable of the node. It is called by: `setMapResponse()`.

*void adjustNodeWeights(aNode,featureVector,gain)* adjusts the node weight-vector in relation to a given feature vector in a proportion defined by gain value, which is given as a parameter. The present solution chooses one component of the feature vector at random, updates it, and then returns. The function branches to treat cases of negative and positive gain separately. It also incorporates a mechanism for limiting the weight adjustment, so as to prevent it from passing the value of the respective feature-vector component – which might happen, especially at higher gain settings. It is called by: `adjustZoneWeights()`.

## Section 4: file `map.h`

*Description: File map.h defines the run-time data structure for the map, as well as for a subset of the map called "zone". It also defines constants, macros, and variables for the map and its zones.*

In the SOMAT system, the overall shape of the map is that of a torus. When it is convenient to do so, however, the map can also be thought of as a two-dimensional rectangular surface. When this surface is folded over so that the upper edge joins the lower one, and the right edge joins the left one, the shape of torus is formed.

In the two-dimensional surface metaphor, the nodes in the extreme left column act as token nodes for each line (horizontal list) of nodes. This column of token nodes (called a *token list*) does not store data, nor is it an active part of the map. For each token node, the state-variable of the node is initialized to the constant `TOKEN`, which is also defined in `map.h`. Moving around on the torus surface is done with four `macros north(Node *aNode), east(Node` `*aNode), south(Node *aNode),` and `west(Node *aNode)`. The purpose of these macros is to describe directions and, more importantly, to pass the token nodes, so that they are never arrived at while browsing through the contents of the map. They are to be thought of as the four main points of compass (north, west, south, and east) in order to remain compatible with the direction pointers of the node links, which were referred to in the description of header file `node.h`.

In addition to the token list, there is one more token node for the entire map, which provides a uniform master access point to the map. It points to the token list with its south pointer, all other direction pointers being `NULL`. The operations on the map are conceived so that they all can be made via this pointer. Bearing this in mind, we note that type-definition of the map is identical to that of a single node, since both `Node` and `Map` are pointers. Certain operations, which deal with the environment of a single node, are needed for processing the map. The concept of *zone* is used for this purpose. Zone is defined as follows:

```
typedef struct {
        Node *centerNode;
        int gain, largeRadius, smallRadius;
} Zone;
```

Typically, the zone around a center node is processed in three stages: The closest neighborhood of a node specializes action similar to that of the center node (positive lateral feedback). A further “ring” around this inner circle specializes in weaker positive-feedback function. In yet a further “ring”, a weak negative lateral feedback is present, which resists the tendencies of the center node.

Macro `maxDistnc(mapSize)` computes the maximum distance between two nodes on a torus surface, when given map-size as a parameter. To define the three-stage neighborhood of a node, three macros are provided to compute the limits of the neighborhood. The width of the neighborhood depends on the size of the map, so the macros are to be passed using global variable `mapSize` as a parameter. Macro `primNeighb(mapSize)` will return the radius of the most immediate neighborhood; `secnNeighb(mapSize)` will return the radius of the intermediate neighborhood; and `tertNeighb(mapSize)` will return the radius of the outermost one. They divide the maximum distance between any two nodes into three equal parts, or to their nearest integer approximation. The constant `MAXDATA` defines the maximum number of elements in the input and output data-arrays.

<u>Section 5</u>: file map.c

*Description: File map.c contains the routines for implementing the map.*

*Map *initMap(mapSize)*
*void normalizeMapState(aMap)*
*int setMapResponse(aMap,featureVector)*
*int getPeakState(aMap)*
*int getBaseState(aMap)*
*int getXitationIndex(aNode)*
*int getPeakXitationIndex(aMap, peakState)*
*Node *getFocusNode(aMap)*
*int focusMap(aMap,responseFocus,dataVector,dataLength)*
*int adjustZoneWeights(aZone,featureVector)*
*int teachFeature(focusNode,featureVector)*
*int trainingCycle(aMap,responseFocus,dataVector,dataLength)*
*int trainMap(aMap,dataVector,dataLength,trainings)*

*Map *initMap(mapSize)* allocates storage for the map with function `initNode()`, joins the pointer connections to torus structure, and sets the token nodes' state-fields to the constant, `TOKEN`. The size of the map is determined by the parameter `mapSize`. This routine also returns a pointer to the map. It is called by: `main()`.

*void normalizeMapState(aMap)* tracks the maximum and minimum state-values in the map, using functions `getPeakState()` and `getBaseState()`. It normalizes all values between 0 and the global parameter `weightRange`. It is called by functions: `trainingCycle()`, `trainMap()`,`displayReaction()`.

*int setMapResponse(aMap,featureVector)* sends one stimulus featurevector to all nodes of the map. It computes the response of the map, and stores the values to the state-variables of the nodes, using function `setNodeResponse()`. It is called by functions: `trainingCycle()`, `focusMap()`,`displayReaction()`.

*int getPeakState(aMap)* returns the maximum value found among the state-fields of the map. It is called by: `normalizeMapState()`, `getFocusNode()`.

*int getBaseState(aMap)* returns the minimum value found among the state-fields of the map. It is called by: `normalizeMapState()`.

*int getXitationIndex(aNode)* returns the so-called *excitation index* for an excitation zone – a quantity describing the product of size and excitation level for a given zone of uniform excitation on the map. The excitation index is defined as the weighted product of two factors: 1) the diameter of a zone of uniform state value – an excitation cluster; 2) the sum of the state-values of all nodes in the zone. It is called by: `getPeakXitationIndex()`.

*int getPeakXitationIndex(aMap, peakState)* returns the maximum excitation index for the map (see the description of `getXitationIndex()`, above, for the definition of excitation index). It is called by: `getFocusNode()`.

*Node *getFocusNode(aMap)* returns a pointer to the node that is returning the maximum excitation index. It is called by: `trainingCycle()`, `displayPeakNode()`, `getVectorCorrelation()`.

*int adjustZoneWeights(aZone,featureVector)* implements lateral feedback for a given zone and feature-vector on the map. The nodes, including and around the small radius but within and exclusive of the large radius, are adjusted as dictated by the gain field of the zone structure. Returns (1). It is called by: `teachFeature()`.

*int teachFeature(focusNode,featureVector)* implements both direct and lateral-feedback teaching for one node of the SOM, which acts as the center node (focus node) for a given feature vector, and its neighborhood. It manages zone specifications for three stages of lateral feedback. In the closest neighborhood region excitation for the feature vector is strengthened with high gain, as intermediate neighborhood implements a weaker strengthening of excitation. In the outer neighborhood the lateral function will be set to slight inhibition. `teachFeature()` calls function `adjustZoneWeights()` to run lateral feedback for center node and each stage of neighborhood. The sizes of the regions are dependent on the size of the map as defined in the header file `map.h` and explained in its documentation. Returns (1). Called by: `trainingCycle()`.

*int trainingCycle(aMap,responseFocus,dataVector,dataLength)* implements one cycle of training for a given data vector; forms display-vectors on the basis of the data vector; sends each feature vector as a stimulus to the map, and lets the map develop a reaction by using function `setMapResponse()`. It calls the function `getFocusNode()` to find the node of highest response value ("focus node") for each of the feature vectors. Foci are used to select zone-center nodes in implementing teaching – both direct and lateral feedback based – by means of the function `teachFeature()`. Returns (1). Called by: `trainMap()`.

*int trainMap(aMap,dataVector,dataLength,trainings)* executes a given number of training cycles with a given data vector by calling the function `trainingCycle()`. It provides bookkeeping of the weight changes made on each cycle. Returns (1). Called by: `main()`.

## Section 6: file wbench.c

*Description: File wbench.c contains the main program. It forms a workbench for testing SOMAT functions, implements a command-line user interface, and makes test runs with real data possible. Additional functions are provided for displaying the results of program runs.*

*int quickRandom(initialization,limit)*
*int readVector(dataVector)*
*int displayData(dataVector,dataLength)*

*int displayMap(aMap)*
*void displayPeakNode(aMap,label)*
*int displayReaction(aMap,dataVector,dataLength)*
*int getVectorCorrelation(aMap,stimulusVector,indexVector)*
*int searchReference(aMap,dataVector,referenceVector,dataLength,k,l, transfer)*
*int classifyDataVector(aMap,referenceVector,dataVector,dataLength)*
*int getMinData(aVector, dataLength)*
*int getMaxData(aVector, dataLength)*
*void scaleDataVector(aVector,dataLength,limit)*
*main(argc,argv)*

Constants `IA`, `IC` and `IM` are used in pseudo-random number generation. Constant `REF_THRESHOLD` is an experimentally chosen ratio used in pattern recognition functions.

*int quickRandom(initialization,limit)* returns a positive-integer random number between 0 and *limit*. A nonzero value for initialization is treated as the seed for a pseudo-random sequence. Called by: `main()`, `initNode()`.

*int readVector(dataVector)* fills the array `dataVector[]` with integers, reading them as formatted text from standard input, and it returns the number of integers read. Called by: `main()`.

*int displayData(dataVector,dataLength)* prints the contents of the array `dataVector` to the standard output. The integers are printed as formatted text, following the direction of the global variable, `outputFormat`. Returns (1). Called by: `main()`.

*int displayMap(aMap)* prints the contents of each state-variable of a map to the standard output row by row. Returns (1). Called by: `displayReaction()`.

*void displayPeakNode(aMap,label)* computes the location of the node of the map in relation to the peak excitation state, and prints it to the standard output. The integer variable *label* can be used to assign a number to a specific node of printout. Called by: `displayReaction()`.

*int displayReaction(aMap,dataVector,dataLength)* computes the reactions of the map to all feature vectors defined by a data vector. The variable dataLength specifies the number of items in the data vector. It prints the reactions of the map as a matrix of state-variable values to the standard output. Returns (1). Called by: `classifyDataVector()`.

*int getVectorCorrelation(aMap,stimulusVector,indexVector)* returns a correlation figure between two vectors computed on the basis of mutual "shadow activation" (as explained earlier) on a trained map. In contradistinction to earlier versions, threshold values are not used in this function, even though the calling function has a threshold mechanism. Called by: `searchReference()`.

*int searchReference(aMap,dataVector,referenceVector,dataLength,k,l,*

*transfer)* implements the back-tracing reference chaining, and marks the detected formal boundaries to a vector called `stimulusVector[],` which records the results of the search, and has the same length as `dataVector[]`. It receives two indices for the `dataVector[]`, in order to cross-reference detection points. As the more stationary of the two, `k` marks a starting point of a feature vector called `stimulusVector[]`. Against this stimulus another vector, called `indexVector[]`, is compared. The starting point of `indexVector[]` is marked by index l, which runs through the whole `dataVector[]` for each `k`. Correlation between stimulus and index vectors is computed by means of the function `getVectorCorrelation()`. If correlation is detected, backwards chaining is then attempted in a recursive loop, with the parameter transfer passing accumulated reference information from each recursion level to the next. When no more correlation is found, or when the beginning of the data vector is reached, the accumulative sum of correlation is written to the respective element of reference vector. The function incorporates a constant `REF_THRESHOLD`, which is the threshold value for interpreting any correlations as an actual reference between elements of `dataVector`[]. If any reference is detected, the function returns (1); if not, it returns (0). Called by: `classifyDataVector()`, self (recursion).

*int classifyDataVector(aMap,referenceVector,dataVector, dataLength)* executes data segmentation on the basis of shadow activation. Returns (1). It takes care of cross-table comparison. Called by: `main()`.

*int getMinData(aVector, dataLength)* returns the minimum values on a data vector. Called by: `scaleDataVector()`.

*int getMaxData(aVector, dataLength)* returns the maximum values on a data vector. Called by: `scaleDataVector()`.

*void scaleDataVector(aVector,dataLength,limit)* scales the values on a data vector between 0 and `limit`. Called by: `main()`.

*main(argc,argv)* is the program that reads data from the standard input, and it creates and teaches a self-organizing map, using functions `initMap()` and `trainMap()`. It produces segmentation of an input data sequence, using the function `classifyDataVector()`, and displays the results, using the functions `displayReaction()` and `displayData()`. The program reads 9 parameters from a separate parameter file, according to the format documentation given in Chapter 8.2.

## 8.4 Program usability and relation to current programming techniques

A further point of interest, as regards SOM and reference chaining, is the following. In principle, it might suitably be applied to analysis of sound, for undoubtedly there is a metric even in time-domain, sound-sample data, which are in some respects comparable to notes in a score, even very simple ones, and are perhaps somewhat easier to detect. The main hindrance to using the system for sound analysis is its heavy demand of computational power. Still, experiments with sound can already be made, by using minimally short sound bursts. An even more restricted area of implementation might be a sequence made up of spectral analysis windows of sound-sample data (frequency domain information).

Future development of the system will involve assigning labels to the phrases of music found through this method of segmentation, and using these labels as source material for a new cycle of analysis. In this way, a higher order of structural hierarchy may be established. At that stage, the challenge will be, that the labeling system should bear a sufficient metric (sufficient amount of iconicity) for SOMAT to allow the abstraction process to take place. It would be easy to provide arbitrary symbolic labels, but that would be contrary to the functionality and nature of SOMAT.

Procedurizing of the semiotic model introduced in Chapters 3 and 5 bears a resemblance to the software technology usually called Object-Oriented Programming (OOP). This resemblance is more than coincidental. It illustrates the closeness of Peircean semiotic theory to sign-processing machines. OOP uses the concept of *class* in a way similar to how we have treated that of *musical object* or *musical legisign*. The term *object* (instead of *class*) was retained in this study for conventional reasons – mainly to retain compatibility with the humanities, instead of concepts from computer science. The OOP term *object* comes very close to what we call *representamen* in our model. The idea of *instantiation* was borrowed directly from the world of OOP, which we found to be in accordance with Peircean semiotics, as was already pointed out. One could easily go on to define equivalents, in the world of musical objects, for *abstract superclasses* and other OOP concepts.

Because OOP is so close to the subject of this study, one may wonder why the programming work was not carried out with the aid of an object-oriented software development package. That might indeed be a natural course of development. It must be noted, however, that a musical legisign is quite likely to be far more complex than the class definitions normally used in OOP applications. More problematic is the fact that legisigns may house ambiguous properties. This, together with portability reasons and a desire to keep things simple and under control, made it more preferable to adopt conventions of traditionally structured functional programming. Nevertheless, the software was written with possible future porting to an OOP environment in mind. Whether or not such an enterprise is undertaken in future, the present study has developed a theoretical framework that might also be usable later.

# 9 Final remarks

On the basis of the experiences with the SOMAT system documented here, one should be cautious in giving too many promises concerning its performance. The attempt was made to tie semiotic and philosophical ideas together into a theoretical construct that is practical enough for computer implementation. Thus, the system was meant to be able to withstand a series of empirical tests, which were constructed following a rigid technological methodology. Hence, if the reader sees worthy achievement in the enterprise documented in this study, part of that achievement is theoretical in nature. In my view, that is the most significant part. On no occasion was it assumed that the main objective of this work would be to develop yet another piece of computer-music software, though it is neither my place nor desire to criticize such useful projects. The subject matter and point of view taken here are simply more theoretical in nature. Nevertheless, it was felt that a real, practical, and empirical series of tests would be needed to inspire and guide the formation of the theory, and to enable closer acquaintance with its interaction with real musical material. In my view, computer-driven empiricism can rarely (if ever) be regarded as an independent and reliable verification of a theoretical construct as such, with the obvious exception of computer science. One should verify or falsify scientific theories by using tools that are not prone to measurement errors or implementation-oriented programming faults or – even worse – faults that derive from the programming platform used in the development of the theory. The value of the kind of empiricism used in this study has been to guide the formation of concepts, and to ensure that any new concepts formed are straightforward and compact enough to serve as cornerstones of functional and applicable systems, and do not remaining "dry academic exercises". Theory and hands-on testing should coexist in harmony in the research and development of new concepts in computer-assisted musicology, even when the work is targeted at studying fundamental principles, and with no interest in developing software tools for producing work of one kind or another.

As a new area of implementation for SOMs, the present study is an example of a growing mass of applications for self-organizing process. It is my hope that this work forms a small contribution to this area, even though its main focus is on music and musicology. In terms of computer science, the present work does contain some new and original material. I would like to think of the idea of *shadow activation* used in SOMAT system as a novel conception. Nevertheless, new ideas rarely appear "out of the blue" without any predecessor, and such this case here, too: the notion of *afterimage* phenomena in self-organizing feature maps came up years ago, in my private discussions with Teuvo Kohonen, though its use in measuring pattern similarity is, if memory serves me, my original idea. I certainly take all responsibility for any plainness and possible lack of elegance regarding its implementation at the level of computer programming. That is to say, possible problems found in any part of the SOMAT system are most likely not related to Kohonen's ideas.

From the evaluation of examples given in Chapter 7, it is clear that the performance of a system like SOMAT can not be fully estimated except in the light of actual musical examples. The use of artificial data can be instructive in testing some special cases of the behavior of the system, but more general observations tend to be of greater validity when the data used are taken from real-world cases. To make such observations requires not only engineering skills, but also significant musical competence.

The software development environment used in this study is a conservative one, to say the least: the work was carried out with programming tools and techniques dating from the 1980s and even the 1970s. On the other hand, from the beginning the purpose was to build the system on a thoroughly tested foundation, instead of running the newest development systems. More flexible tools are already at hand, and no doubt much more developed ones will appear in the future. The overcoming of technical problems will perhaps become easier in the future, at least in some respects.

As concerns computer implementations of Peircean semiotic theory, I see my own contribution as one possible starting point, rather than as a final word of any kind. What this study has clearly shown, is that the branch of semiotics started by Charles Sanders Peirce, and the increasing body of its musical applications that have appeared recently, will indeed also be able to support computer-assisted musical studies, even in the largely machine-oriented context of algorithm development. Moreover, it seems evident that Peircean semiotics continues to be a rich source for theoretical frameworks. It has the capacity to bring together diverse ideas from diverse fields of inquiry, which was my main target when beginning work on this study. From this, we may conclude that the original work and thought, which went into building the foundations of Peirce's semiotics, have stood the test of time, and have done so very well.

# References

Agawu, V. Kofi (1991). *Playing with Signs: A Semiotic Interpretation of Classic Music*. Princeton, NJ: Princeton University Press.

Alexander, Igor and Helen Morton (1990). *An Introduction to Neural Computing*. London: Chapman & Hall.

Austin, William W. (1966). *Music in the 20th Century*. New York: Norton.

Backus, John (1969). *The Acoustical Foundations of Music*. New York: Norton.

Baker, Michael (1989). A computational approach to modeling musical grouping structure. *Contemporary Music Review* 4: 311-325.

Baroni, Mario (1983). The concept of musical grammar. *Music Analysis* 2.2: 175-208.

Baroni, Mario et al. (1982). A grammar for melody: Relationships between melody and harmony. In: M. Baroni and Laura Callegari (eds.), *Musical Grammars and Computer Analysis*. 201-218. Florence: Olschki.

Bent, Ian (1987). *Analysis*. London: Macmillan.

Bharucha, Jamshed J. (1987). Music cognition and perceptual facilitation: A connectionist framework. *Music Perception* 5.1: 1-30.

— (1991). Pitch, harmony and neural nets: A psychological perspective. In: Peter M. Todd and D. Gareth Loy (eds.), *Music And Connectionism*. 84-99. Cambridge, MA: MIT Press.

Bharucha, Jamshed J. and Katherine L. Olney (1989). Tonal cognition, artificial intelligence and neural nets. *Contemporary Music Review* 4: 341-356.

Boethius. *Fundamentals of Music* (1989). Tr. Calvin M. Bower. Ed. Claude V. Palisca. New Haven, CT: Yale University Press.

Broeckx, Jan L. and Walter Landrieu (1972). Comparative computer study of style, based on five liedmelodies. *Interface* 1: 29-92.

Cambouropoulos, Emilios (1996a). Musical rhythm: A formal model for determining local boundaries, accents and metre in a melodic surface. In: Marc Leman (ed.), *Music, Gestalt and Computing*. 277-293. Berlin: Springer-Verlag.

— (1996b). A formal theory for the discovery of local boundaries in a melodic surface. In: *Proceedings of the Troisièmes Journées d'Informatique Musicale (JIM-96)*. Caen: GREYC – Université de Caen.

Carpenter, G. A. and Stephen Grossberg (1987). ART 2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics* 26: 4919-4930.

Clarke, Eric F. (1989). Mind the gap: Formal structures and psychological processes in music. *Contemporary Music Review* 3.1: 1-13.

Cohen, Joel E. (1962). Information theory and music. *Behavioral Science* 7.2: 137-163.

Dowling, W. Jay and David L. Harwood (1986). *Music Cognition*. Orlando, FL: Academic Press.

Ebcioglu, Kemal (1988). An expert system for harmonizing four-part chorales. *Computer Music Journal* 12.3: 43-51.

Forte, Allen (1967). Computer-implemented analysis of musical structure. In: Gerald Lefkoff (ed.), *Computer Applications in Music*. Morgantown, WV: West Virginia University.

Frydén, Lars and Johan Sundberg (1984). Performance rules for melodies: Origin, functions, purposes. In: *Proceedings of the International Computer Music Conference*. 221-224. San Francisco, CA: Computer Music Association.

Gjerdingen, Robert O. (1991). Using connectionist models to explore complex musical patterns. In: Peter Todd and D. Gareth Loy, *Music And Connectionism*. 138-149. Cambridge, MA: MIT Press.

Granger, Gilles-Gaston (1968). *Essai d'une philosophie du style*. Paris: Armand Colin.

Greenlee, Douglas (1973). *Peirce's Concept of Sign*. The Hague: Mouton de Gruyter.

Grossberg, Stephen (1976). Adaptive pattern classification and universal recording II: Feedback, expectation, olfaction, and illusions. *Biological Cybernetics* 23: 187-202.

Hecht-Nielsen, Robert (1991). *Neurocomputing*. Reading, MA: Addison-Wesley.

Hofstadter, Douglas R. (1979). *Gödel, Escher, Bach: An Eternal Golden Braid.* New York: Basic Books.

Holtzman, Steven R. (1980). A generative grammar definition language for music. *Interface* 9: 1-48.

Holtzman, Steven R. (1994). *Digital Mantras: The Languages of Abstract and Virtual Worlds*. Cambridge, MA: MIT Press.

Klír, Jirí and Miroslav Valach (1967). *Cybernetic Modelling*. London (UK): Iliffe Books. Original pub. Prague: SNTL, 1965.

Kohonen, Teuvo (1988). The "neural" phonetic typewriter. *Computer* (IEEE) 21.3: 11-22.

— (1989). *Self-Organization and Associative Memory*. 3rd ed. Berlin: Springer-Verlag.

— (2001). *Self-Organizing Maps*. 3rd ed. Berlin: Springer-Verlag.

Kohonen, Teuvo, Pauli Laine, Kalev Tiits, and Kari Torkkola (1991). A nonheuristic automatic composing method. In: Peter M. Todd and D. Gareth Loy (eds.): *Music and Connectionism*. 229-242. Cambridge, MA: MIT Press.

Kraehenbuehl, David and Edgar Coons (1959). Information as a measure of the experience of music. *Journal of Aesthetics and Art Criticism* 17.4: 510-522.

Lane, Michael (1970). *Structuralism: A Reader*. London: Jonathan Cape.

Laine, Pauli (2000). *A Method for Generating Musical Motion Patterns*. Diss., University of Helsinki.

Laske, Otto E. (1975). Introduction to a generative theory of music: Part 2. *Sonological Reports No. 1b.* Utrecht: Institute of Sonology.

— (1988). Introduction to cognitive musicology. *Computer Music Journal* 12.1: 43-57.

Leman, Marc (1991a). *Symbolic and Sub-symbolic Description of Music*. (= Seminar of Musicology and Institute for Psychoacoustics and Electronic Music, Report 20.) Ghent: University of Ghent.

— (1991b). *Tone Context and the Complex Dynamics of Tone Semantics*. (= Seminar of Musicology and Institute for Psychoacoustics and Electronic Music, Report 22.) Ghent: University of Ghent.

— (1992). Some epistemological considerations on symbolic and subsymbolic processing. In: *Actes du Colloque Musique et Assistance Informatique.* 35-50. Marseille: Laboratoire Musique et Informatique de Marseille.

Leman, Marc and Francesco Carreras (1997). Schema and gestalt: Testing the hypothesis of psychoneural isomorphism by computer simulation. In: M. Leman (ed.), *Music, Gestalt and Computing.* 13-29. Berlin: Springer-Verlag.

Leman, Marc and Albrecht Schneider (1997). Origin and nature of cognitive and systematic musicology: An introduction. In: M. Leman (ed.), *Music, Gestalt and Computing.* 13-29. Berlin: Springer-Verlag.

Lerdahl, Fred and Ray Jackendoff (1983). *A Generative Theory of Tonal Music.* Cambridge, MA: MIT Press.

Lewin, David (1968). Some applications of communication theory to the study of twelve-tone music. *Journal of Music Theory* 12.1: 50-84.

Lindblom, Björn and Johan Sundberg (1970). Towards a generative theory of melody. *Swedish Journal of Musicology* 52: 71-88.

Lischka, Cristoph (1987). Connectionist models of musical thinking. *Proceedings of the International Computer Music Conference.* Urbana, IL: Computer Music Association.

— (1991). Understanding music cognition: A connectionist view. In: Giovanni de Poli et al. (eds.), *Representations of Musical Signals.* 417-445. Cambridge, MA: MIT Press.

Loy, D. Gareth (1991). Connectionism and musiconomy. In: Peter Todd and D. Gareth Loy (eds.), *Music and Connectionism.* 229-242. Cambridge, MA: MIT Press.

Maxwell, Harry John Jr. (1984). *An Artificial Intelligence Approach to Computer-Implemented Analysis of Harmony in Tonal Music.* Diss., Indiana University-Bloomington.

Martinez, Jose Luiz (1997). *Semiosis in Hindustani Music*. (= Acta Musicologica Fennica 5.) Imatra, Finland: International Semiotics Institute.

Meehan, James R. (1980). An artificial intelligence approach to tonal music theory. *Computer Music Journal* 4.2: 60-65.

Mesarovic, Mihajlo D (1962). On self organizational systems. In: Marshall C. Yovits et al. (eds.), *Self-Organizing Systems 1962*. 9-36. Washington, D.C.: Spartan Books

Meyer, Leonard B. (1956). *Emotion and Meaning in Music*. Chicago, IL: University of Chicago Press.

Meyer, Leonard B. (1967). *Music, the Arts and Ideas*. Chicago, IL: University of Chicago Press.

Mirigliano, Rosario (1995). The sign and music: A reflection on the theoretical bases of musical semiotics. In: Eero Tarasti (ed.), *Musical Signification: Essays in the Semiotic Theory and Analysis of Music*. Berlin: Mouton de Gruyter.

Moles, Abraham (1966). *Information Theory and Esthetic Perception*. Urbana, IL: University of Illinois Press.

Monelle, Raymond (1992). *Linguistics and Semiotics in Music*. Chur, Switzerland: Harwood Academic Publishers.

Morris, Charles (1971). Foundations of the theory of signs. In: Otto Neurath, Rudolph Carnap, and Charles Morris (eds.), *Foundations of the Unity of Science 1*. 3rd ed. 78-137. impression. Chicago, IL: University of Chicago Press.

Murre, Jacob Marinus Jan (1992). *Categorization and Learning in Neural Networks: Modelling and Implementation in a Modular Framework*. Diss., University of Leiden.

Nattiez, Jean-Jacques (1973). Linguistics: A new approach for musical analysis. *International Review of the Aesthetics and Sociology of Music* 4.1: 51-68.

Neumann, John von (1966). *Theory of Self-Reproducing Automata*. Ed. and completed posthumously by Arthur W. Burks. Urbana, IL: University of Illinois Press.

Peirce, Charles S. (1936-1958). *Collected Papers of Charles Sanders Peirce*. Ed. Charles Hartshorne et al. Cambridge, MA: Harvard University Press. [Referred to as Peirce [volume number].[paragraph number].]

— (1955). *Philosophical Writings of Peirce.* Ed. Justus Buchler. New York: Dover.

Pinkerton, R. C. (1956). Information theory and melody. *Scientific American* 194: 77-86.

Ritter, Helge (1991). Learning with the self-organizing map. In: *Artificial Neural Networks: Proceedings of the 1991 International Conference on Artificial Neural Networks*. 379-384). Amsterdam: North-Holland.

Roads, Curtis (1980). Artificial intelligence and music. *Computer Music Journal* 4.2: 13-25.

— (1982). On overview of music representations. In: Mario Baroni and Laura Callegari (eds.), *Musical Grammars and Computer Analysis.* 7-37. Florence: Olschki.

— (1985). Grammars as representations of music. In: C. Roads and John Strawn (eds.), *Foundations of Computer Music*. 403-442. Cambridge, MA: MIT Press.

Schaeffer, Pierre (1966). *Traité des objets musicaux*. Paris: Seuil.

Schillinger, Joseph (1978). *The Schillinger System of Musical Composition*. New York: Da Capo Press.

Schalkoff, Robert (1992). *Pattern Recognition: Statistical, Structural and Neural Approaches*. New York: John Wiley & Sons.

Schottstaedt, William (1989). Automatic counterpoint. In: Max V. Mathews and John R. Pierce (eds.), *Current Directions in Computer Music Research*. 199-214. Cambridge, MA: MIT Press.

Shannon, Claude E. and Warren Weaver (1949). *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press.

Sharma, Prem Lata (1970). Rasa theory and Indian music. *Sangeet Natak Academi* 16: 57-64.

Slawson, Wayne (1968). Review of G. Lefkoff (ed.), *Computer Applications in Music*. *Journal of Music Theory* 12.1: 105-111.

Smalley, Denis (1986). Spectro-morphology and structuring processes. In: Simon Emmerson (ed.), *The Language of Electroacoustic Music*. 61-93. Houndmills, UK: Macmillan.

Smolensky, Paul (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences* 11: 1-74.

Smoliar, Stephen W. (1973). Basic research in computer-music studies. *Interface* 2: 121-125.

Tarasti, Eero (1978). *Myth and Music*. (= Acta Musicologica Fennica 11.) Helsinki: Finnish Musicological Society.

— (1994). [Private communication.]

Tenney, James and Larry Polansky (1987). Temporal gestalt perception in music. *Journal of Music Theory* 24.2: 205-241.

Tempelaars, Stan (1992). *Signal Processing, Speech and Music*. Den Haag: Edition Koninklijk Conservatorium.

Thomas, Marily Taft (1985). *Vivace*: A rule based AI system for composition. In: *Proceedings of the International Computer Music Conference*. 267-274. San Francisco, CA.

Tiits, Kalev (1992). Automatic analysis of melody: A computer implementation. In: *Actes du Colloque Musique et Assistance Informatique*. 347-353. Marseille: Laboratoire Musique et Informatique de Marseille.

— (1994). *The Formalized Process of Abstraction in Segmenting Analysis of Music*. Thesis, Licenciate of Philosophy, University of Helsinki.

Todd, Peter (1989). A connectionist approach to algorithmic composition. *Computer Music Journal* 13.4: 27-43.

Winograd, Terry (1968). Linguistics and the computer analysis of tonal harmony. *Journal of Music Theory* 12.1: 2-49.

Xenakis, Iannis (1971). *Formalized Music*. Bloomington, IN: Indiana University Press.

— (1985). Music composition treks. In: Curtis Roads (ed.), *Composers and the Computer*. 171-192. Los Altos, CA: William Kaufmann, Inc.

# Appendix 1 SOMAT source code listing

```
/* VERSION 1.1 © Kalev Tiits 2001 */
/* the code includes some commented debugging and MS-DOS compile options */


/* file somat.h: global parameter structure */

#define MAXDATA 5000 /* length of input & output data arrays */
#define MAXERROR 10 /* error bookkeeping for development */

typedef struct {
   Node *focusVector[MAXDATA];
   int seed, trainings, spaceDimension, mapSize;
   int outputFormat, weightsChanged, errorCount;
   int weightRange, lateralGain[3];
   int verbose, bidirectionalProcess;
} ParamStruct;

ParamStruct globalPara;


/* file node.h: SOMAT node interface */

/* define MSDOS */
#ifdef MSDOS
#include <malloc.h>
#endif

#define MAXDIMENSION 10
#define NULL 0

typedef struct nd {
   int weightVector[MAXDIMENSION];
   int state;
   struct nd *n, *w, *s, *e; /* n,w,s,e are the 4 main directions */
} Node;

#include "somat.h"
Node *initNode();
int setNodeWeights(), setNodeResponse(), adjustNodeWeights();
```

```
/* file node.c: SOMAT node implementation */

#include "node.h"

Node *initNode()
{
 extern ParamStruct globalPara;
 Node *aNode;
 int k;
 aNode = (Node*)malloc(sizeof(Node));
 aNode->n = aNode->w = aNode->s = aNode->e = NULL;
 aNode->state = 0;
 for (k = 0 ; k < MAXDIMENSION ; k++)
     aNode->weightVector[k] = quickRandom(0,globalPara.weightRange);
 return(aNode);
}

setNodeResponse(aNode,featureVector)
Node *aNode;
int featureVector[];
{
 extern ParamStruct globalPara;
 int k, sum;
 sum = globalPara.spaceDimension * globalPara.weightRange;
 for (k = 0 ; k < globalPara.spaceDimension ; k++)
     sum-= abs(aNode->weightVector[k] – featureVector[k]);
 aNode->state = sum;
}

adjustNodeWeights(aNode,featureVector,gain)
Node *aNode;
int featureVector[], gain;
{
 extern ParamStruct globalPara;
 int k, difference, increment = 0;
 /* one of the weights chosen at random for updating */
 k = quickRandom(0,(globalPara.spaceDimension – 1));
 difference = aNode->weightVector[k] – featureVector[k];
 if (gain > 0) {                /* case of exitatory adjustment */
    if (difference > 0)          /* if decrease of weight needed */
       increment = (-1) * ((difference < gain)? difference : gain);
    else if (difference < 0) {    /* if increase of weight needed */
       difference*= (-1);
       increment = ((difference < gain)? difference : gain);
    } /* else if */
 }
 if (gain < 0) {                /* case of inhibitory adjustment */
    if (difference > 0) {        /* if increase of weight needed  */
```

```
      gain*= (-1);
      increment = ((aNode->weightVector[k] + gain) < globalPara.weightRange)?
           gain : (globalPara.weightRange – aNode->weightVector[k]);
   } /* if */
   /* the following case also includes the situation of exact match */
   else if (difference <= 0)      /* if decrease of weight needed   */
      increment = ((aNode->weightVector[k] + gain) > NULL)?
                 gain : ((-1) * aNode->weightVector[k]);
 } /* if */
 if (increment != 0) {
   aNode->weightVector[k]+=increment;
   globalPara.weightsChanged++;
 }
}


/* file map.h: SOMAT map interface */

#include "node.h"
#include <math.h>
#define TOKEN (-1)

/* two ways to define maximum distance of two arbitrary nodes are possible
 on the rectangular map topology; one is called 'city-block distance' */

#define maxDistnc(n) ((int)((float)n/2))

/* other way to measure distances is the familiar Euclidean distance */

/*#define maxDistnc(n) (1 + (int)floor(sqrt(2.0) * (((float)n)/2)))*/

/* Since the zone definitions in SOMAT versions 1.x use rectangular zones, city-
block distance metric is used. If the program is later updated for circular zones,
Euclidean distances and respectively the latter macro for maxDistnc would be
appropriate. For lateral functions three-stage neighbourhood limits  are defined in
respect to maximum node distance on the chosen map topology : */

#define primNeighb(n) (maxDistnc(n) / 3)
#define secnNeighb(n) (2 * maxDistnc(n) / 3)
#define tertNeighb(n) maxDistnc(n)

#define east(a)  ((a->e->state == TOKEN) ? a->e->e : a->e)
#define west(a)  ((a->w->state == TOKEN) ? a->w->w : a->w)
#define north(a) ((a->n->state == TOKEN) ? a->n->n : a->n)
#define south(a) ((a->s->state == TOKEN) ? a->s->s : a->s)

typedef Node Map;
typedef struct {
        Node *centerNode;
        int gain, largeRadius, smallRadius;
} Zone;

Map *initMap();
Node *getFocusNode();
int normalizeMapState(), setMapResponse(), getPeakState(), focusMap(),
adjustFeedbackZone(), executeLateralFunction(), trainingCycle(), trainMap();


/* file map.c: SOMAT map implementation */

#include "map.h"
Map *initMap(mapSize)
int mapSize;
{
 Map *aMap;
 Node *rowNode, *colNode, *newNode, *tokenNode;
 int i, j;
 aMap = colNode = initNode();
 for (i = 0 ; i < mapSize ; i++) {
     /* next 4 lines make a new token node */
     newNode = initNode();
     newNode->n = colNode;
     colNode->s = newNode;
     colNode = rowNode = newNode;
     for (j = 0 ; j < mapSize ; j++) {
         newNode = initNode();
         newNode->w = rowNode;
         rowNode->e = newNode;
         rowNode = newNode;
         if (i > 0) {
            newNode->n = newNode->w->n->e;
            newNode->n->s = newNode;
         }
      }
 }
 /* joining and marking of the token nodes */
 for (colNode = aMap->s ; colNode->s ; colNode = colNode->s)
     colNode->state = TOKEN;
 colNode->state = TOKEN;
 colNode->s = aMap;
 aMap->n = colNode;
 aMap->state = TOKEN;
 /* joining the east and the west edges: */
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s) {
     for (rowNode = colNode->e ; rowNode->e ; rowNode = rowNode->e);
```

```
    rowNode->e = colNode;
    colNode->w = rowNode;
 }
 /* joining the north and the south edges: */
 tokenNode = aMap->s;
 for (rowNode = tokenNode->e ; rowNode != tokenNode ; rowNode = rowNode->e) {
    for (colNode = rowNode ; colNode->s ; colNode = colNode->s);
     colNode->s = rowNode;
     rowNode->n = colNode;
 }
 return(aMap);
}

normalizeMapState(aMap)
Map *aMap;
{
 Node *rowNode, *colNode;
 int peak, base;
 peak = getPeakState(aMap);
 base = getBaseState(aMap);
 peak = peak > 0 ? peak : 1;  /* divide by 0 precaution */
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s)
    for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e)
       rowNode->state = ((rowNode->state – base) *
                         globalPara.weightRange) / (peak – base);
 return(1);
}

setMapResponse(aMap,featureVector)
Map *aMap;
int featureVector[];
{
 Node *rowNode, *colNode;
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s)
    for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e)
        setNodeResponse(rowNode,featureVector);
 return(1);
}
```

```
getPeakState(aMap)
Map *aMap;
{
 Node *rowNode, *colNode;
 int peak = 0;
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s)
     for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e)
         if (rowNode->state > peak)
            peak = rowNode->state;
 return(peak);
}

getBaseState(aMap)
Map *aMap;
{
 Node *rowNode, *colNode;
 int base = 10000;
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s)
     for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e)
         if (rowNode->state < base)
            base = rowNode->state;
 return(base);
}

getXitationIndex(aNode)
Node *aNode;
{
 extern ParamStruct globalPara;
 int k, state, workDistance, activity, xitationIndex, searchTerminate = 0;
 Zone aZone;
 Node *workNode;
 aZone.centerNode = aNode;
 state = aNode->state;
 for (workDistance = 0 ; ((!searchTerminate) &&
   (workDistance < maxDistnc(globalPara.mapSize)));workDistance++) {
     /* next 3 lines move workNode to upper left corner of the working area */
     workNode = aZone.centerNode;
     activity = 0;
     for (k = 0 ; k++ < workDistance ; workNode = north(workNode));
     for (k = 0 ; k++ < workDistance ; workNode =  west(workNode));
     /* next 8 lines process nodes in a region around centerNode */
     for (k = 0 ; k++ < (workDistance * 2) ; workNode = south(workNode)) {
         if (workNode->state != state) searchTerminate = 1;
         activity+= workNode->state;
     }
     for (k = 0 ; k++ < (workDistance * 2) ; workNode =  east(workNode)) {
         if (workNode->state != state) searchTerminate = 1;
         activity+= workNode->state;
     }
     for (k = 0 ; k++ < (workDistance * 2) ; workNode = north(workNode)) {
        if (workNode->state != state) searchTerminate = 1;
        activity+= workNode->state;
     }
     for (k = 0 ; k++ < (workDistance * 2) ; workNode =  west(workNode)) {
        if (workNode->state != state) searchTerminate = 1;
        activity+= workNode->state;
     }
 } /* value of function is a combination of radius of uniform state
      and its excitation level */
 xitationIndex = 300 * workDistance + activity;
 return(xitationIndex);
}

getPeakXitationIndex(aMap, peakState)
Map *aMap;
int peakState;
{
 Node *rowNode, *colNode;
 int index, peak = 0;
 extern ParamStruct globalPara;
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s)
     for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e)
        if (rowNode->state == peakState) {
           index = getXitationIndex(rowNode);
           if (index > peak)
              peak = index;
        }
 if (peak == 0) {
    printf("WARNING: getPeakXitationIndex returns 0\n");
    displayMap(aMap);
    globalPara.errorCount++;
    if (globalPara.errorCount > MAXERROR)
       exit(0);
 }
 return(peak);
}

Node *getFocusNode(aMap)
Map *aMap;
{
 Node *rowNode, *colNode, *peakNode;
 int xitationIndex, peakState, peakXitationIndex;
 peakState = getPeakState(aMap);
 peakXitationIndex = getPeakXitationIndex(aMap,peakState);
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s)
     for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e)
```

```
        if (rowNode->state == peakState) {
           xitationIndex = getXitationIndex(rowNode);
           if (xitationIndex ==  peakXitationIndex) {
              peakNode = rowNode;
              break;
           }
        }
 return(peakNode);
}

adjustZoneWeights(aZone,featureVector)
Zone aZone;
int featureVector[];
{
 Node *workNode;
 int k, workDistance;
 workNode = aZone.centerNode;
 /* adjust weights for centerNode */
 if (aZone.smallRadius == 0) {
    adjustNodeWeights(workNode,featureVector,aZone.gain);
    aZone.smallRadius++;
 }
 /* cover the neighbourhood up to the radius */
 for (workDistance = aZone.smallRadius ; workDistance < aZone.largeRadius ;
     workDistance++) {
     /* next 3 lines move workNode to upper left corner of the working area */
     workNode = aZone.centerNode;
     for (k = 0 ; k++ < workDistance ; workNode = north(workNode));
     for (k = 0 ; k++ < workDistance ; workNode =  west(workNode));
     /* next 8 lines process nodes in a 'circle' around focusNode */
     for (k = 0 ; k++ < (workDistance * 2) ; workNode = south(workNode))
        adjustNodeWeights(workNode,featureVector,aZone.gain);
     for (k = 0 ; k++ < (workDistance * 2) ; workNode =  east(workNode))
        adjustNodeWeights(workNode,featureVector,aZone.gain);
     for (k = 0 ; k++ < (workDistance * 2) ; workNode = north(workNode))
        adjustNodeWeights(workNode,featureVector,aZone.gain);
     for (k = 0 ; k++ < (workDistance * 2) ; workNode =  west(workNode))
        adjustNodeWeights(workNode,featureVector,aZone.gain);
 }
 return(1);
}
```

```
teachFeature(focusNode,featureVector)
Node *focusNode;
int featureVector[];
{
 extern ParamStruct globalPara;
 Zone aZone;
 aZone.centerNode = focusNode;
 aZone.gain = globalPara.lateralGain[0];
 aZone.smallRadius = 0;
 aZone.largeRadius = primNeighb(globalPara.mapSize);
 adjustZoneWeights(aZone,featureVector);
 aZone.gain = globalPara.lateralGain[1];
 aZone.smallRadius = primNeighb(globalPara.mapSize);
 aZone.largeRadius = secnNeighb(globalPara.mapSize);
 adjustZoneWeights(aZone,featureVector);
 aZone.gain = globalPara.lateralGain[2];
 aZone.smallRadius = secnNeighb(globalPara.mapSize);
 aZone.largeRadius = tertNeighb(globalPara.mapSize);
 adjustZoneWeights(aZone,featureVector);
 return(1);
}

trainingCycle(aMap,dataVector,dataLength)
Map *aMap;
int dataLength, dataVector[];
{
 extern ParamStruct globalPara;
 Node *focusNode;
 int k, l, featureVector[MAXDIMENSION];
 for (k = 0 ; k < (dataLength – (globalPara.spaceDimension -1)) ; k++) {
     for (l = 0 ; l < globalPara.spaceDimension ; l++)
         featureVector[l] = dataVector[(k + l)];
     setMapResponse(aMap,featureVector);
     focusNode = getFocusNode(aMap);
     teachFeature(focusNode,featureVector);
 }
 return(1);
}
```

```
trainMap(aMap,dataVector,dataLength,trainings)
Map *aMap;
int dataVector[], dataLength, trainings;
{
 extern ParamStruct globalPara;
 int k;
 for (k = 0 ; k < trainings ; k++) {
   globalPara.weightsChanged = 0;
   trainingCycle(aMap,dataVector,dataLength);
   if (globalPara.verbose)
      printf("cycle %4d – chg %5d \n",(k+1),globalPara.weightsChanged);
 }
 return(1);
}



/* file wbench.c: SOMAT map main program and experimentation workbench */


#include "map.h"
#include <stdio.h>

#define IA 577L
#define IC 29L
#define IM 12759L
#define REF_THRESHOLD (29 * globalPara.weightRange / 30)
#define MAX_VALUE 32500

quickRandom(initialization,limit)
int initialization, limit;
{
 static long seed;
 int rnd;
 if (initialization)
    seed = (long)initialization;
 seed = (seed * IA + IC) % IM;
 rnd = (int)((long)limit + 1) * seed / IM;
 return(rnd);
}

readVector(dataVector)
int dataVector[];
{
 int k;
 for (k = 0 ; ((scanf("%d",&dataVector[k]) == 1) && (k < MAXDATA)) ; k++);
 return(k);
}
```

```
displayData(dataVector,dataLength,outputFormat)
int dataVector[], dataLength, outputFormat;
{
 int k;
 for (k = 0 ; k < dataLength ; printf("%7d ",dataVector[k++]))
    if (!(k % outputFormat))
       printf("\n");
 return(1);
}

displayMap(aMap)
Map *aMap;
{
 Node *rowNode, *colNode;
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s) {
    for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e)
       printf("%2d ",rowNode->state);
    printf("\n");
 }
 printf("\n\n");
 return(1);
}

displayPeakNode(aMap,label)
Map *aMap;
int label;
{
 Node *rowNode, *colNode, *focusNode;
 int row, col, peak;
 row = 1;
 focusNode = getFocusNode(aMap);
 for (colNode = aMap->s ; colNode != aMap ; colNode = colNode->s) {
    col = 1;
    for (rowNode = colNode->e ; rowNode != colNode ; rowNode = rowNode->e) {
       if (rowNode == focusNode)
          printf("Map %d, Focus row %d column %d\n",label,row,col);
       col++;
    }
    row++;
 }
}
```

```
displayReaction(aMap,dataVector,dataLength)
Map *aMap;
int dataVector[], dataLength;
{
 extern ParamStruct globalPara;
 int featureVector[MAXDIMENSION], k, m;
 printf("\n\n\n");
 for (k = 0 ; k < (dataLength – (globalPara.spaceDimension – 1)) ; k++) {
     for (m = 0 ; m < globalPara.spaceDimension ; m++)
         featureVector[m] = dataVector[(m + k)];
     setMapResponse(aMap,featureVector);
     displayPeakNode(aMap,(k + 1));
     normalizeMapState(aMap);
     displayMap(aMap);
 }
 return(1);
}

getVectorCorrelation(aMap,stimulusVector,indexVector,k,l)
Map *aMap;
int stimulusVector[], indexVector[], k, l;
{
 Node *stimulusFocus, *indexFocus;
 extern ParamStruct globalPara;
 int shadowActivation = 0;
 stimulusFocus = globalPara.focusVector[k]; /* resp. focus for stim. */
 indexFocus = globalPara.focusVector[l];    /* resp. focus for index */
/* first compute shadow of stimulus to index excitation */
 setMapResponse(aMap,indexVector);
 normalizeMapState(aMap);
 shadowActivation+= stimulusFocus->state;
/* then compute shadow of index to stimulus excitation */
 setMapResponse(aMap,stimulusVector);
 normalizeMapState(aMap);
 shadowActivation+= indexFocus->state;
/* final activation – mean of both values */
 shadowActivation/= 2;
 return(shadowActivation);
}
```

```
/* the ref. chaining implemented in a recursive loop in searchReference */


searchReference(aMap,dataVector,referenceVector,dataLength,k,l,transfer)
Map *aMap;
int dataVector[], referenceVector[], dataLength, k,l,transfer;
{ /* chaining of the references */
 extern ParamStruct globalPara;
 int stimulusVector[MAXDIMENSION], indexVector[MAXDIMENSION], m,
     correlationStrength = 0;
 for (m = 0 ; m < globalPara.spaceDimension ; m++) {
     stimulusVector[m] = dataVector[(m + k)];
     indexVector[m] = dataVector[(m + l)];
 }
 correlationStrength = getVectorCorrelation(aMap,stimulusVector,indexVector,k,l);
 if (correlationStrength > REF_THRESHOLD) {
    transfer+= correlationStrength;
    if ((k == 0) || (l == 0)) /* beg. of dataVector as a special case */
      referenceVector[k]+= transfer;
    else  /* chaining in a recursive loop */
      if (searchReference(aMap,dataVector,referenceVector,dataLength,--k,--
l,transfer) == 0)
         referenceVector[++k]+= transfer;
    return(1);
 }
 else
   return(0);
}

/*       cross-reference detection done with two indices k and l.  k determines the
         feature being detected and l runs through all elements of dataVector except
         the one indexed by k.  */

classifyDataVector(aMap,referenceVector,dataVector,dataLength)
Map *aMap;
int referenceVector[], dataVector[], dataLength;
{
 extern ParamStruct globalPara;
 int k, l, status;
 for (k = 0 ; k < (dataLength – (globalPara.spaceDimension – 1)) ; k++) {
     for (l = (k == 0 ? 1 : 0) ;
      l < (dataLength – (globalPara.spaceDimension – 1)) ;
      l+= (l != (k – 1)) ? 1 : 2) {
        searchReference(aMap,dataVector,referenceVector,dataLength,k,l,0);
     }
 }
 return(1);
}
```

```
getMinData(aVector, dataLength)
int aVector[], dataLength;
{
 int k, min = MAX_VALUE;
 for (k = 0 ; k < dataLength ; k++)
     if (aVector[k] < min)
        min = aVector[k];
 return(min);
}

getMaxData(aVector, dataLength)
int aVector[], dataLength;
{
 int k, max = 0;
 for (k = 0 ; k < dataLength ; k++)
     if (aVector[k] > max)
        max = aVector[k];
 return(max);
}

scaleDataVector(aVector,dataLength,limit)
int aVector[], dataLength, limit;
{
 int k, minData, maxData;
 minData = getMinData(aVector, dataLength);
 maxData = getMaxData(aVector, dataLength);
 for (k = 0 ; k < dataLength ; k++)
     aVector[k] = limit * (aVector[k] – minData) / (maxData – minData);
}

focusMap(aMap,dataVector,dataLength)
Map *aMap;
int dataLength, dataVector[];
{
 extern ParamStruct globalPara;
 int k, l, featureVector[MAXDIMENSION];
 for (k = 0 ; k < (dataLength – (globalPara.spaceDimension – 1)) ; k++) {
     for (l = 0 ; l < globalPara.spaceDimension ; l++)
         featureVector[l] = dataVector[(k + l)];
     setMapResponse(aMap,featureVector);
     globalPara.focusVector[k] = getFocusNode(aMap);
 }
 return(1);
}
```

```
getParameters(s)
char *s;
{
 extern ParamStruct globalPara;
 FILE *in, *fopen();
 char str[120];
 int n;
 if (fopen(s,"r"))
    in = fopen(s,"r");
 else {
    printf("Parameter file not found\n");
    exit(1);
 }
 for (n = fscanf(in,"%s",str) ; n == 1 ; n = fscanf(in,"%s",str)) {
     if (!strcmp(str,"seed"))
        n = fscanf(in,"%d",&globalPara.seed);
     else if (!strcmp(str,"trainings"))
        n = fscanf(in,"%d",&globalPara.trainings);
     else if (!strcmp(str,"spaceDimension"))
        n = fscanf(in,"%d",&globalPara.spaceDimension);
     else if (!strcmp(str,"mapSize"))
        n = fscanf(in,"%d",&globalPara.mapSize);
     else if (!strcmp(str,"outputFormat"))
        n = fscanf(in,"%d",&globalPara.outputFormat);
     else if (!strcmp(str,"weightRange"))
        n = fscanf(in,"%d",&globalPara.weightRange);
     else if (!strcmp(str,"lateralFunctionGains")) {
        n = fscanf(in,"%d",&globalPara.lateralGain[0]);
        n = fscanf(in,"%d",&globalPara.lateralGain[1]);
        n = fscanf(in,"%d",&globalPara.lateralGain[2]);
     }
     else if (!strcmp(str,"verbose"))
        n = fscanf(in,"%d",&globalPara.verbose);
     else if (!strcmp(str,"bidirectionalProcess"))
        n = fscanf(in,"%d",&globalPara.bidirectionalProcess);
     else if ((str[0] != '/') || (str[1] != '*')) {
        printf("Error in parameter file: %s\n",str);
        exit(1);
     }
 }
}

initDataVector(aVector)
int aVector[];
{
 int k;
 for (k = 0 ; k < MAXDATA ; aVector[k++] = 0);
}
```

```
reverseDataVector(dataVector,dataLength)
int dataVector[], dataLength;
{
 int k, buffer[MAXDATA];
 for (k = 0 ; k < dataLength ; k++)
     buffer[k] = dataVector[k];
 for (k = 0 ; k < dataLength ; k++)
     dataVector[k] = buffer[(dataLength-1-k)];
}


main(argc,argv)
int argc;
char *argv[];
{
 Map *aMap;
 extern ParamStruct globalPara;
 int k, ran, seed, trainings, dataLength, outputFormat,
     dataVector[MAXDATA], referenceVector[MAXDATA];
 if (argc < 2) {
    printf("usage %s parameterFile\n",argv[0]);
    printf(" – parameterFile contains the following parameters:\n");
    printf("     seed\n     trainings\n     spaceDimension\n");
    printf("     mapSize\n     outputFormat\n     weightRange\n");
    printf("     lateralFunctionGains (3 values)\n");
    printf("     verbose\n     bidirectionalProcess\n");
    exit(0);
 }
 getParameters(argv[1]);
 if (globalPara.spaceDimension > MAXDIMENSION)
    globalPara.spaceDimension = MAXDIMENSION;
 if (globalPara.mapSize < 6)
    globalPara.mapSize = 6;
 globalPara.errorCount = 0;
 quickRandom(globalPara.seed,10); /* initialize random generator */
 aMap = initMap(globalPara.mapSize);
 initDataVector(referenceVector);
 dataLength = readVector(dataVector);
 scaleDataVector(dataVector,dataLength,globalPara.weightRange);
 trainMap(aMap,dataVector,dataLength,globalPara.trainings);
 focusMap(aMap,dataVector,dataLength);/* bookkeeping of resp. foci est. */
 classifyDataVector(aMap,referenceVector,dataVector,dataLength);
 if (globalPara.bidirectionalProcess) {
    if (globalPara.verbose)
       displayReaction(aMap,dataVector,dataLength);
    reverseDataVector(dataVector,dataLength);
    reverseDataVector(referenceVector,dataLength);
    trainMap(aMap,dataVector,dataLength,globalPara.trainings);
    focusMap(aMap,dataVector,dataLength);/* bookkeeping of resp. foci est. */
    classifyDataVector(aMap,referenceVector,dataVector,dataLength);
 }
if (globalPara.bidirectionalProcess) {
    if (globalPara.verbose)
       displayReaction(aMap,dataVector,dataLength);
    reverseDataVector(dataVector,dataLength);
    reverseDataVector(referenceVector,dataLength);
 }
 displayData(referenceVector,dataLength,globalPara.outputFormat);
}
```