The Symbolic vs. Numeric Controversy in Automatic Analysis of Music

Rémy Mouton
Laboratoire Son & Vision,
Université Paris I, 162 rue Saint-Charles,
75016 Paris, France
Email: mouton@clipper.ens.fr

François Pachet
Laforia-IBP, Université Paris 6, Boîte 168,
4, Place Jussieu, 75252 Paris Cedex
France
Email: pachet@laforia.ibp.fr

Abstract

This position paper describes two families of approaches to automatic analysis of tonal music: approaches based on purely numerical computations, and approaches based on symbolic models. Both approaches have been used in our team with great success, but they address different levels of difficulties, and aim at analyzing different corpus of musical material. Indeed, they are difficult to reconcile and use in a single collaborative environment. We claim that finding ways to integrate smoothly both approaches is the only real bottleneck to producing reliable automatic harmonic analyzers.

1. Introduction

Humdrum [Huron, 1994] is a set of general-purpose software tools developed at university of Waterloo (Canada), intended to assist researchers in posing and answering research questions. Humdrum is one of the most ambitious attempts so far at providing computational power to musicians in order to perform complex analysis of musical pieces. In the Faq file of the Humdrum package, one can read the following question/answer: "Can Humdrum do automatic roman-numeral type harmonic analysis ?", answer is: "Not currently. Programs to do automatic functional analysis are not sufficiently reliable to be used in music scholarships". This position paper aims at giving insights on the real limitations of current research in automatic analysis of tonal music.

We are interested in building reliable automatic analysis software systems (as opposed to tool kits). This motivation is not a challenge made by a team of ambitious computer scientists. Rather, we think that only automatic analysis of a large corpus of tonal music may eventually provide us (the human) with insights on the very nature of tonal music. Among our goals, we are interested for example in: finding sets of rules governing the structure and shape of "well-balanced" melodies, uncover rules governing musical styles of Bach or Mozart, or rules expliciting the judicious choice of modulations. Our ultimate goal is to write a modern treatise on harmonization and orchestration that can be directly used by students to help them have a solid grasp on tonal music without having to spend years studying musical corpus.

Analyzing music automatically is a challenging task for Artificial Intelligence for many reasons : musical material is made up of a great variety of information (timbre, pitches, rhythm, dynamics, harmony) ; unlike image, music is a linear (temporal) process. An obvious analogy is to compare it with verbal speech, without the complexity of the reference to some material *signifié* . Finally, some parts of the repertoire, from Bach chorals to Real Book be-bop tunes, make up well defined subsets, thereby allowing exhaustive empirical studies.

Research works on automatic musical analysis use different approaches of the phenomenon corresponding to different lines of action : the choice of analyzing directly a sound document, a MIDI file (i.e. the result of a performance), a score (that contains all information regardless of interpretation), or an harmonic grid, greatly influences the choice of a *method* as well as the choice of the *features* of the resulting programs, such as the possibility or obligation for the user to intervene during the analysis process or the ease he'll have to correct its results.

This paper first shortly reviews the major researches on automatic analysis of tonal music at a higher level than the rough physical signal level. Some of these works try to extract some *higher level information* from the chosen input data (e.g. ciphering harmonies from a score) ; others aim at *detecting regularities* in the data (e.g. forming a set of grammar rules for a corpus, yielding statistical regularities, ...). It is believed here that these two types of analysis cooperate and complement each other. A full synthesis of these works in currently is progress [Pachet, 1995]. The emphasis here will be put on two important dimensions of research on analyzing automatons that have never been successfully combined: 1) *numeric* tools, based on measurements performed on the data followed by the use of thresholds for making decisions, and 2) *symbolic* tools, that use abstract concepts and relations together with symbolic inferences. We will discuss the various pros and cons of these two kinds of approaches, and the ways to lead them to cooperation, according to our experience in this field.

2. Major researches in automatic musical analysis

Computer analysis of tonal music has been tackled by virtually all available computer formalisms and methodologies : grammar-based, constraint-based, frame-based, network based. Historically, there has been a strong tradition in computer music to try to find grammar-based models that capture essential aspects of the deep structures of the music being analyzed. The proposed models owe much to the work of [Lerdahl and Jackendoff, 1983], who crystallized a long tradition of efforts to adapt or transpose linguistic theories, and especially Chomskian theories, to the musical domain. As Lerdahl and Jackendoff themselves mention, these models are not intended to provide directly an implementation model or a set of algorithms. Indeed, several later efforts have concentrated in trying to implement the grammar-based models, with more or less limited success (see e.g. [Deliège, 1994].

However, all the models proposed are not standard generative models. [Roads, 1988] reviews grammar-based approaches for analysis and generation purposes, and shows how different grammar models account for various musical tasks, emphasizing on the relative difficulties of implementing these models, particularly models containing so-called context-dependent rules. Among these approaches, [Steedman, 1984] provides an elegant grammar for 12-bar blues. His grammar contains context-free as well as context-dependent rules. The mere presence of context-dependent rules makes his model not suitable for implementation, and therefore can only be useful in a "contemplative mode". Other approaches for finding grammars to be used for tonal music include [Sundberg and Lindbloom, 1993], and [Olshki, 1984]. [Laske, 1993] argues that grammars should be seen mainly as explications of musical competence.

Based on this discrepancy between 1) the elegance and lack of workability of grammar-based models and 2) the need of building effectively systems for empirical studies, several attempts have been made to build systems that perform some kind of harmonic analysis automatically. In a first category, systems aim at providing tools for computer-assisted analysis. This is typically the case of [Byrd, 1977], [Brinkman, 1980] and [Smoliar, 1980]. Because we are interested only in analyzing large corpus of musical material automatically, tool kits are not appropriate.

The second category of systems tries to produce complete analysis of musical material without human intervention. Real complete analysis is much more difficult than mere tabulation of data about surface features. [Rothgeb, 1968] was the first to tackle the "Figured bass" problem, i.e. provide harmonizations of figured bass lines. As he points out in [Rothgeb, 1989], the most tangible result of this work seems to be the definitive proof that classical treatises are largely insufficient and under-specified : "General solutions to the unfigured-bass problem were probably inaccessible to procedures of the type represented by those of Heinichen and Saint-Lambert". In the purely analytical domain, [Winograd, 1993] used Augmented Transition Networks (an extended grammar formalism) to implement analysis of Bach chorales. [Winold and Bein, 1983] use a standard artificial intelligence approach to tackle the same problem. Inspired by these works, [Maxwell, 1992] proposes an expert system in the form of a set of production rules to perform harmonic chord function analysis. His system includes 55 rules organized in three levels of control (or meta rules). Other approaches have been used for similar tasks: [Ulrich, 1977] analyses jazz harmonies, with an "island-growing" approach that has an appealing organic quality. [Steels, 1979] uses a constraint-based approach to reason about tonal structures in music, and shows how to solve the "passing-chord problem" using constraints, by inserting a chord that is harmonically "near" its predecessor and successor. Finally, [Meehan, 1980] uses conceptual dependency graphs to implement the implication/realization theory of [Narmour, 1977] .

3. Two systems for AHA

We conducted some research on automatic analysis of tonal music for several years in two directions. First, a research is in progress that uses statistical and procedural methods to produce reliable automatic analyzers for large corpus of tonal music, mainly taken from the classical repertoire. This work addresses a series of problems such as: 1) a *pitch speller* that reconstitutes enharmonic spelling from context, given, e.g. a rough MIDI file, 2) a *style recognizer* for complex orchestrations (e.g., detects "Alberti bass" from a non-annotated score), a pure harmonic analyzer (detects underlying tonalities in a score), leading to a harmonic ciphering system, a *regularity extractor* for tonal material [Mouton, 1995]. Second, we developed a model and a system that produces correct harmonic analysis of jazz chord sequences as found in the Real Book or Fake Book series. This model uses purely symbolic methods taken from object-oriented knowledge representation, production rules, and declarative control architecture [Pachet, 1991]. Both approaches share a common goal: be able to produce reliable, fully operating systems that produce acceptable analysis for most of the corpus analyzed. We will now describe each approach more in detail.

3.1. The NUSO system

The NUSO system aims at providing musicologists with a set of automatic tools for analyzing tonal music.

It mainly addresses the classical, non-improvised, written musical corpus.

The first feature NUSO provides is an automatic translator from MIDI files into an analytic notation. Such a translator has to find the spelling of enharmonic notes (like G# and Ab), that share the same key on the keyboard and thus the same MIDI number. The correct spelling for a note depends on several considerations, such as: the ambient tonality and the roles the note plays in the harmony and in the melody. At this stage of the analysis, NUSO just produces a "first guess" of the spelling based on an estimation of the ambient tonality. To estimate the tonality at each point of the piece, we first use several successive *filters* that count the occurrences of each pitch class in the MIDI file. Then the tonality is estimated by considering the local frequency of the pitch classes, in view of some basic assumptions on these frequencies in tonal music. These assumptions are:

- "the most frequent notes are the tonic and the dominant",
- "the most frequent chords are the triad of the tonic and the seventh chord on the dominant",

and so forth.
Finally, each note is given an unique spelling in the most likely tonality at its location.

Some further analyzing features use or will use the same kind of filtering methods. Comparing the number of different pitches, the number of different pitch classes, and the number of notes struck by each instrument within a lapse of time gives an idea of the orchestration style (parallel chords with or without doubling of notes, parallel octaves, Alberti-style melodic bass accompanying a melody, ...). The frequencies of the pitch classes obtained by the previous filters, along with the use of weights depending on the rhythmic and orchestral position of each note, lead to chord detection and harmonic ciphering. The weights are computed in relation with the orchestration style detected, the style of the piece (baroque, romantic, ...) and some statistic assumptions :

- "in the case of an accompanied melody, the notes of the melody are more likely to be ornamental (non-harmonic) than the notes of the accompaniment" (see figure 2),
- "in the case of rich, complex parallel chords, it is likely that each stroke can be analyzed as a chord (as opposed to an Alberti bass where the chords are spread into arpeggios)",
- "in the baroque style, the seventh of the dominant can be struck at the same time as the chord only if the leading note is at the bass",

etc.

A threshold gives the possibility of adapting dynamically the width of the filtering window.

A third step of the analyzing process by NUSO detects the regularities in different texts extracted from the music by the musicologist user (helped by suggestions of NUSO). For instance, providing this regularity detector with the list of the intervals of a melody - the alterations being ignored in this context - will show the repetitions in this melody, with possible changed mode and/or tonality. Again, statistic assumptions such as:

- "a theme is longer than the pattern of a harmonic march",
- "the pattern of a development is usually extracted from a theme", etc.,

along with the previously estimated harmonic ciphering, lead to the recognition of themes, patterns and conclusive formulas. The different ornamentations of a pattern are detected by the regularity detector if it is provided with a list of the "important notes" of the piece, or the intervals between them. A selection of the "important notes" is proposed by NUSO according (again) to statistic assumptions, such as:

- "notes that are part of the chords are more important than melodic ones",
- "long notes are more important than short ones", etc.

(see figure 1).

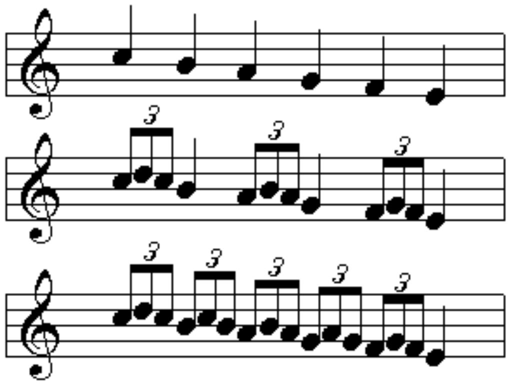

Figure 1. Detecting ornementations of a given motif.

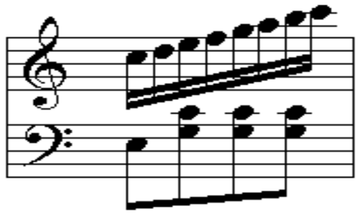

Figure 2. Finding the underlying tonality of an accompanied melody.

3.2. The MusES system

The aim of the MusES system is to build up a model for the analysis of jazz chord sequences, as found in the standard corpus of [Real, 1981], or [Fake, 1983; 1991]. Our goal is similar in spirit to the goal of the NUSO system in that we want our model to be fully operational, and account for most of the regularities found in this corpus. The very nature of the corpus however, calls for utterly different techniques as is described here.

The problem of jazz chord sequence analysis consists in computing, for a given chord sequence (as the one in Figure 5). the underlying tonality of each of its chords. The main characteristics of this analysis is that it is hierarchical: a tune may be globally in C major, but some parts of it may be in F (modulation), and so on. Generally speaking, harmonic analysis produces a tree with which each chord of the sequence may be analyzed, at several levels of abstractions. Figure 6 shows one possible analysis tree for the tune in Figure 5.

Lastly, the aim of the analysis is usually to provide, for each chord of the sequence, indications to the musicians for improvisation. These indications are the underlying tonalities (at all levels of abstractions), as well as identifications of well-known "patterns" that make sense for the improviser, because he will be able to use pre-defined licks well adapted to these patterns.

The theory behind, revisited

Like classical harmony, tonal jazz harmony is a well studied domain, as one can see by browsing at the numerous books written on this subject [Beaudoin, 1990; Coker, 1964]. However, few books attempt at providing a model for the analytic process per se. The situation is actually comparable to the situation in linguistics : if lots of works have attempted to find grammars for natural languages, only few operational models of language understanding have been developed.

Before describing our model for analysis, we propose to formalize the problem around three major points, as follows:

A) Basic principles
The theory is based on two major principles:
1) A "legality" principle
This principle says that each chord, out of any context, can be analyzed in a fixed set of possible tonalities. A tonality is faithfully represented as a scale (a list of notes) and a degree. For instance, a C major chord may be analyzed as: I st degree of C major scale, IVth degree of G major, Vth of F major, VI de E harmonic minor, and so forth. Note that the computation of this "legal set" is entirely deterministic.
2) A minimization principle
In a context, the choice of the "good" tonality for a chord will of course depend on its location, and its relation with adjacent chords. The main idea here is that the best tonality will be the one that minimizes modulations, i.e. that is common to the greatest number of adjacent chords. For instance, the sequence (C / F / E min / A min) has only one tonality that is common to all chords: C major.

B) Perturbations
This nice and simple theory is complicated a phenomenons that escapes rigorous formalization, but which are essential to capture the essence of the process: substitutions and idioms.

First, some chords may be substituted by others, and the substitute often violates the legality principle. For instance, a seventh chord that resolves may be substituted by its tritone seventh (C7 -> F#7). Second, there are a number of well-known idiomatic "musical shapes" that bear particular harmonic meaning in themselves. This is the case of "two-fives", turnarounds, and other similar shapes. These shapes are remarkable in that they may be analyzed out of their context. Thus, the sequence "Cmaj7/A 7/Dmin7/Db7" is in itself a turnaround in C major, regardless of the fact that C major does not belong to the legal set of Db7. In other terms, Db7 *in abstracto* may not be analyzed in C major, and can only be within such a musical shape.

C) Recursion
Lastly, the process is *recursive*. This means that any recognized shape may itself be considered as atomic for a higher level of analysis. This recursive nature accounts for the hierarchical nature of the analysis. For instance, resolving seventh chords may be considered as preparations, and therefore integrated to their resolving chord. Typically, the sequence: "A7 / D7 / G7 / C" may be entirely analyzed in C major, thanks to a recursive reasoning (see Figure 3).

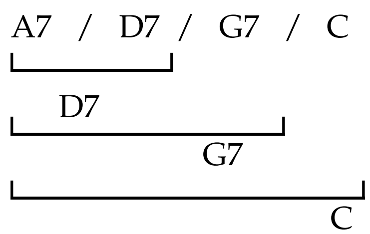

Figure 3. The hierarchical nature of the analysis of a group of chords.

At the highest level, global macro forms are introduced in a similar fashion. Thus, a Blues is identified by a succession of 3 musical shapes, covering 12 bars, and such that the fourth of the root of the middle one's tonality is equal to the root of the first and of the last shape. Structures such as AABA or ABAB may be described similarly.

The analysis reasoning in MusES

Our approach is based on an explicit reconstruction of the reasoning process. This process is represented by 1) a rich representation of the concepts used during the reasoning, and 2) a representation of the reasoning by production rules.

The objects used during the analysis reasoning are basically the objects making up the universe of discourse, i.e. the chords and the chord sequence, together with more abstract objects that represent the various shapes and tonalities identified during the reasoning. These abstract objects are represented in a conceptual hierarchy (see Figure 7).

The reasoning process per se is represented by a series of rule bases, that basically perform two kinds of tasks:
- a "pattern recognition" task

in which higher level shapes are identified from configurations of lower level shapes.
- a "forgetting" task

in which irrelevant or redundant shapes are destroyed.

Pattern recognition rules all follow the same pattern:

```
FOR a1 a2 instances of AnalysableObject
IF some harmonic properties between a1 and a2 are
satisfied
THEN
Create an instance of a particular class (subclass of
Shape) covering the durations of a1 and a2.
Establish the composition link between the newly
created shape and a1 and a2.
```

For instance, here is the rule (in a Smalltalk-like pseudo syntax) that recognizes a two-five in major (such as "Dmin7/G7"):

```
majorTwoFive
FOR a1 a2 instances of AnalysableObject
IF
   c1 isMinor.
   (c1 hasA: #flatFifth) not.
   c2 isAfter: c1.
   c2 isMajor.
   c2 hasA: #minorSeventh.
   c2 root pitchEqual: c1 root fourth.
THEN
Create a TwoFive, x.
   x beginBeat: c1 beginBeat; endBeat: c2 endBeat.
   x tonality: (c2 root fourth majorScale);
Establish composition lik between x and c1 and c2 .
```

Other rules describe shapes such as resolutions (A7 / D), turnarounds, and substitutions. More abstract rules describe more complex phenomenons such as "modal borrowing": a local modulation may be considered as non significant in certain cases, when it comes in between two shapes analyzable in the same tonality (Cf. Figure 4).

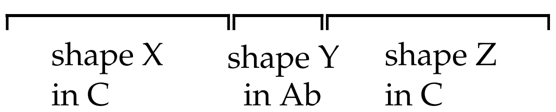

Figure 4. Modal borrowing configuration.

Forgetting rules
The second type of rules describe typical situations in which recognized shapes may be safely forgotten, to speed up the reasoning process, and avoid combinatorial explosion. Such rules include rule `removeSubsumedShapes`, that allows forgetting a shape safely without loosing information, when it is subsumed by another shape of the same tonality:

```
removeSubsumedShapes
FOR c1 c2 instances of ObjectInASequence
IF
   c1 subsumes: c2.
   c1 ~= c2.
   c2 tonality notNil.
   c1 tonality notNil.
   c1 tonality  = c2 tonality.
THEN
   chordSequence removeAnalysis: c2
```

The overall reasoning is therefore represented as a series of rule bases alternating shape recognition and shape forgetting. The precise scheduling of tasks in described in details in [Pachet, 1995b]. At the end of the reasoning, the complete analysis tree is produced.

The system is now in the evaluation phase, and already proved capable of analyzing blues chord sequences deemed difficult by [Steedman, 1984].

4. Why combining ?

Each approach has proved particularly well suited to specific tasks, and terribly awkward for other. Here are two typical examples of this discrepancy.

For instance, finding the most probable underlying tonality of the sequence of notes in Figure 2 is an easy task for a system that computes statistical weights. Simply computing the frequency of each notes shows that the most frequent notes of the sequence are C, E, G. Based on the assumption that the root, fifth and third are the most frequently used notes for a given tonality, and based on the fact that these notes are on strong metric positions, inferring that the sequence is in C is then trivial.

In the Jazz chord analyzer, the model is based on "shape recognition" scheme that produces a tree of analysis with several levels of abstraction. The leaves of the tree are the chords of the sequences. Intermediary nodes represent well known harmonic shapes, such as "two-fives", turnarounds, or substitutions. Higher-level nodes represent more abstract shapes such as Blues, ABAB, or AABA. The system used production rules combined with objects to detect shapes on multiple levels. The system has proved particularly well adapted to the be-bop tune corpus. For instance, the sequence of chords: "Cmaj7/Eb7/Abmaj7/Dbmaj7" is easily recognized as an instance of a turnaround in C major, possibly with some substitution of seventh chord.

However, it is clear that such a model is utterly incapable of solving the first problem. Conversely, be-bop tunes are often so twisted by multiple substitutions and harmonic tricks that the use of statistics based on the relative frequency of roots, fifths and thirds does not suffice to uncover the actual harmonic intentions of the grid.

5 . Discussion: thresholds vs. rules

Two important remarks must precede the comparison of numeric and symbolic methods. The first is, that the only *pure* numeric systems are neural networks. In any other system, if the system performs more than just counting the occurrences of some totally defined pattern, some rules must exist to decide *what* to measure and *where* to measure it according to the expected result and according to information already obtained. For instance, some rule must say that if you found a cadence, then you should look backwards in the sequence for a possible preparation for the cadence [Mouton, 1994].

The second remark is that both kinds of information, numeric and symbolic, can be useful and interesting to perform analysis. However, symbolic information is usually much more valuable, because it is more synthetic. For instance, saying that the "leading note usually goes up to the tonic", is more efficiently represented in such terms, than with statistics and numbers as "in 90% of cases, B is followed by C". This is not only because rules are less flexible than real numbers. The difference is deeper as pointed out by [Lenat, 1992]. Indeed, stating that the leading note leads up to the tonic is a statement that stands by itself, and does not induce any commitment on, say, the way "thirds would descend a fourth to the seventh". On the contrary, representing such a rule with numbers induce a total order that is misleading, both for the human user and for the system. In a way, saying it with figures implies too many things that are simply not wanted. For instance, if a system finds out that " in 60% of cases, the second leads up to the third" and "in 55 % of cases the sixth goes down to the fifth"; the same system would implicitly assume that "second leads up to the third" is more probable than "the sixth goes down to the fifth", and make the corresponding decisions during the computation, which is simply wrong and dangerous.

In many cases, however, symbolic rules are mainly justified by statistics : in the foreword of a good treatise on harmony, it will generally be written that the rules of the treatise should always be implicitly preceded by a modifier such as "in most cases" or "if there is no particular intention". Therefore the numeric result can be a useful information, even without thresholding or making any decision.

The first problem encountered by computer analysis is to find a *trade-off* between modeling and implementabiliy. The systems that give a good model for the phenomenon they study (a corpus or a human competence) are usually not easy to implement. On the other hand, the systems that really perform the task of analyzing music have seldom satisfactory explanatory power. This can look like an advantage for the rule-based systems, because each rule of such a system is a piece of symbolic knowledge that is supposed to bear some semantics in itself. On the contrary, it is very difficult to interpret a neural network. In practice, the problem of interpretation is to be found in both approaches : a system that uses *ad hoc* or too many rules is not more explanatory than a system based on thresholding : the synthetic information looked for can be hidden behind a heavy inference engine as easily as behind numerization.

Once solved the problem of implementation comes the question of the *reliability* of an analyzing automaton. The basic rules of a rule system are perfectly reliable, because they give a yes or no answer about the presence of such or such figure in the data. However, the whole system can be very sensitive to *rare cases* and perturbed by them [Steedman, 1984]. It can also be unreliable when it contains incompatible or competing rules. A threshold system is intrinsically not perfectly reliable, but the distance to the threshold gives a simple - and reliable - evaluation of the reliability of the results. The difficulty there is to propagate and combine such an evaluation along several analyzing processes. For example, a Midi file may be spelled out with a resulting probability of p1, with will then be used by a ciphering process, that adds up a probability of p2. Although fuzzy logic may provide some answer to this question, the resulting probability is not clear in terms of the musical process involved.

The comparison of the two systems is particularly enlightening on the following points:

1) Ornementations

In the NUSO system, there are an infinite number of possible ornementations (examples of figure 1 could be expanded at will), but each ornamentation is relatively close to the original: filtering methods may exploit the stability of the variations. In the MusES system, substitutions play the role of ornamentation. As we saw, substitutions introduce violent "illegal" tonalities (epitomized by the ubiquitous tritone substitution). But this instability is compensated by the fact that these substitutions are not infinite, and can be easily listed once for all. In this case, an explicit and symbolic representation of the analytic process is not only possible, but necessary.

2) Temporal models of the musical world

It is important to note the importance of the representation of temporal structures underlying each model. In MusES particularly, the temporal model plays a crucial role. This is most visible in the problem of linking ends to beginning of pieces. In jazz chord sequence, there is an implicit assumption that the end of a sequence turns back to its beginning. We discovered that it is extremely important to represent this property of chord sequences explicitly in the model. It allows to explain some difficult modulations in sequences such as: (C/C7/F/..../G 7), where the first chord (here C) should be analyzed as a first degree, followed by a modulation in F. In fact, the analysis of C as first degree of C is possible only if the end of the piece (G 7) is connected to the beginning, thereby strengthening the C major tonality of C for the first chord. Otherwise, since C is analyzable in F, the C chord is "incorporated" in the tonality of F !

In the NUSO system, the situation is quite the opposite. Not only classical pieces do not end by un-resolving sevenths (so the problem of connecting ends to beginnings is irrelevant), but moreover, once can safely make the assumption that endings of tunes actually resolve in the main tonality. This assumption may be used by the automatic analyzer safely to infer quickly the overall tonality of a piece, whereas at the beginning of the analysis, whereas the information of the overall tonality is only deduced at then end of the reasoning process in MusES.

3) Bootstrap hypothesis
There are a number of cases where some circularity in the analytical process are to be found. For instance, in NUSO, the estimation of the tonality of a piece is used by the orthographier module itself leading to the ciphering module, which in turn may be used to infer the tonality. Similarly, the detection of a melodic repetition can lead to the detection of a harmonic repetition, itself leading to the detection of a melodic repetition. Finally, the estimation of the style (e.g. orchestral) may be used to infer the bass, which in turns gives indications on the style.

In MusES, these circularities are represented in the form of conflicts in the rule inference cycle, and therefore represented by conflict resolution strategies.

6. Conclusion : towards a cooperation

There are classical ways to mix numeric and symbolic methods in AI. Rules can be numerized by "fuzzyfying" the logic they are based on, or symbolic information forced out of a numeric system by giving a name to each threshold. The comparisons of section 3, along with our own experience, allow us to give some guidelines.

In a general manner, numerizing some parameter comes down to giving up any *grammatical meaning* for that parameter. The rules within a rule system reach their limit of refinement when it comes to numbers. So the natural way to make numeric and symbolic systems cooperate is to let upper-level symbols rule the organization and use of lower-level procedural, numeric features. These two layers, numeric and symbolic, are not always separated in the way the automaton is implemented, but, in order for the system to keep some explanatory power, the constructor should take care of the possibility of examining them separately when interpreting the system once it is run. This, indeed, reflects the way a neurologist can look at a brain : either as a set of cells that exchange numeric data, or as a set of zones that interact to perform logical, symbolic operations.

In practice, if rules can stay together in a jumble, we found difficulties to make numeric processes communicate between different levels in an analysis. A numeric process does not easily take into account information obtained previously, and does not perform "feedback" deductions easily. In the case of harmonic analysis, for instance, an estimation of the tonality of a given passage can lead to a change in the ciphering of the underlying chords, but this change of ciphering can in turn lead to a different analysis of the tonality. We found it difficult in NUSO to let each step of the analyzing process be able to take into account some information computed by a further level of the analysis, because each level computes a specific kind of information, that eventually uses a specific notation, that is meaningful for an eventually different scale of time or a different grouping of the events (chords, melodies, ...). In fact, numeric methods force to *segment* the task, because they are always low-level, even if they apply to the results of some previous high-level computation or if they count the applications of the rules of some other system. So the explanation given by a set of numeric processes is to be looked for in the way these processes constrain the segmentation of the task, and in the resulting significant categories, more than in the way the task is performed. Such a segmentation by numeric layers can also avoid a rule system to be too bushy to give any explanation.

A general direction that may provide hints as to when to use which approach is to remark that the symbolic approach is well adapted to musical problems for which there exists some discourse on the abstract shapes to be conceived during the reasoning. In the jazz chord sequence problem, such shapes are two-fives, harmonic series, and tune structures. When no such discourse exists - which is the case for the analysis of arbitrary sequences of notes - then only statistical methods together with strong assumptions on the frequencies of notes can be used. Therefore, the present situation looks like a dilemma: for "shapeless" material, numerical methods are quite appropriate, but then there is a feedback or recursion problem. When the domain allows a reasonable reification of analysis objects - such as for jazz chord sequences - the recursion problem may be avoided, but the approaches requires a totally explicit model, i.e. there can be dangerous "holes" in the rule systems that are not easily filled.

7. References

Brinkman, A.R., Johann Sebastian Bach's Orgelbüchlein. *Music Theory Spectrum.* 2 pp. 46-73. 1980.

Byrd, D., An Integrated Computer Music Software System. *Computer Music Journal* 1 pp. 55+60. 1977.

Deliège, Session on "Lerdahl & Jackendoff : 10 years on, in *International Conference on Music Perception and Cognition (ICMPC).* Liège (Belgium), 1994.

Fake, *The World's Greatest Fake book*. ed. C. Sher. 1983, San Franscisco: Sher Music Co.

Fake, *The New Real Book*. ed. C. Sher. Vol. 2. 1991, Petaluma: Sher Music Co.

Laske, O., In search of a Generative Grammar for Music, in *Machine models of Music*, S.M. Schwanauer and D.A. Levitt, Editor. 1993, MIT Press: p. 215-240.

D. Lenat, R.V. Guha. Building large knowledge-Based Systems. Representation and Inference in the Cyc project. *Addison-Wesley*, 1990.

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

Maxwell, H.J., An Expert System for Harmonizing Analysis of Tonal Music, in *Understanding Music with AI: Perspectives on Music Cognition*, K.Ebcioglu, O.Laske and M. Balaban, Editors. 1992, AAAI Press: p. 335-353.

Meehan, J., An Artificial Intelligence Approach to Tonal Music Theory. *Computer Music Journal* 4 (2), pp. 61-65. 1980.

Mouton, R. Automating the Music Scholar Thinking Process. Ph.D. Thesis, University of Paris I. To be published.

Narmour, E., Beyond Schenkerism. 1977, University of Chicago Press.

Olshki, O., Musical Grammars and Computer Analysis. In *Grammars and Music*, M. Baroni & al ed. 1984, Florence, Italy.

Pachet, F. A meta-level architecture for analysing jazz chord sequences. Proceedings of International Conference on Computer Music, pp. 266-269, Montréal. 1991.

Pachet, F. Musical Analysis: a Synthesis. Laforia-IBP technical report, to appear. 1995.

Pachet, F. Analysis of jazz chord sequences: the MusES approach. Laforia-IBP technical report, to appear. 1995b.

Real, *The Real Book*. 1981, The Real Book Press.

Roads, C., Grammars as Representations for Music, in *Foundations of Computer Music,* C. Roads and J. Strawn, Editor. 1988, MIT Press: p. 403-442.

Rothgeb, J., Harmonizing the Unfigured Bass: a Computational Study, Ph.D. thesis, Indiana University, 1968.

Rothgeb, J., Simulating Musical Skills by Digital Computer, in *The Music Machine*, C. Roads, Editor. 1989, MIT Press: p. 657-661.

Smoliar, S.W., A Computer Aid for Schenkerian Analysis. *Computer Music Journal* 4 (2), pp. 41-59. 1980.

Steedman, M.J., A Generative Grammar for Jazz Chord Sequences. *Music Perception*. 2 (1), pp. 52-77. 1984.

Steels, L., Reasoning modeled as a Society of Communicating Experts, MIT AI Lab., n. AI-TR-542, 1979.

Sundberg, J. and B. Lindbloom, Generative Theories in Language and Music Descriptions, in *Machines Models of Music* (reprint), S.M. Schwanauer. and D.A. Levitt, Editor. 1993, MIT Press: p. 263-286.

Winograd, T., Linguistic and Computer Analysis of Tonal Harmony, in *Machines Models of Music* (reprint), S.M. Schwanauer and D.A. Levitt, Editor. 1993, MIT Press: p. 113-153.

Winold, A. and J. Bein, BANALYSE: An Artificial Intelligence System for Harmonic Analysis of Bach Chorales, Indiana University, Unpublished manuscript, 1983.

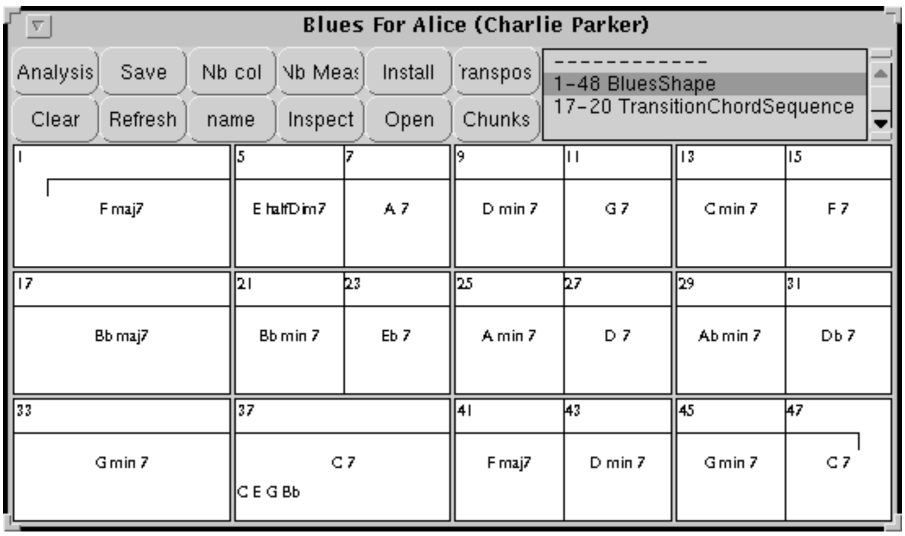

Figure 5. A typical jazz chord sequence to be analysed by MusES.

1-48 BluesShape in F MajorScale
45-4 ResolvanteEndToBeginning in F MajorScale
45-48 ChordSubstitution in F MajorScale
45-48 TwoFive in F MajorScale
[G #min 7]
[C 7]
[F #maj7]
5-20 ChordSubstitution in Bb MajorScale
5-20 Resolvante in Bb MajorScale
5-16 ChordSubstitution in Bb MajorScale
5-16 TwoFive in Bb MajorScale
5-14 ChordSubstitution in C HarmonicMinor

```
          5-14 Resolvante in C HarmonicMinor
            5-12 ChordSubstitution in C MajorScale
              5-12 TwoFive in C MajorScale
                5-10 ChordSubstitution in D HarmonicMinor
                  5-10 Resolvante in D HarmonicMinor
                    5-8 ChordSubstitution in D HarmonicMinor
                      5-8 TwoFive in D HarmonicMinor
                        [E #halfDim7 ]
                        [A 7 ]
                    [D #min 7 ]
                [G 7 ]
            [C #min 7 ]
        [F 7 ]
    [Bb #maj7 ]
17-48 ExtendedShape in F MajorScale
  17-46 ExtendedShape in F MajorScale
    17-44 ExtendedShape in F MajorScale
      [Bb #maj7 ]
      21-44 ExtendedShape in F MajorScale
        21-42 ChordSubstitution in F MajorScale
          21-42 Resolvante in F MajorScale
            21-40 ChordSubstitution in C MajorScale
              21-40 Resolvante in C MajorScale
                21-32 ChordSubstitution in Db MajorScale
                  21-32 Resolvante in Db MajorScale
                    21-28 ChordSubstitution in D MajorScale
                      21-28 Resolvante in D MajorScale
                        21-24 ChordSubstitution in Ab MajorScale
                          21-24 TwoFive in Ab MajorScale
                            [Bb #min 7 ]
                            [Eb 7 ]
                        25-28 ChordSubstitution in G MajorScale
                          25-28 TwoFive in G MajorScale
                            [A #min 7 ]
                            [D 7 ]
                    29-32 ChordSubstitution in Gb MajorScale
                      29-32 TwoFive in Gb MajorScale
                        [Ab #min 7 ]
                        [Db 7 ]
                33-40 ChordSubstitution in F MajorScale
                  33-40 TwoFive in F MajorScale
                    [G #min 7 ]
                    [C 7 ]
            [F #maj7 ]
        [D #min 7 ]
    [G #min 7 ]
  [C 7 ]
```

Figure 6. The complete analysis of the chord sequence of Figure 5.

```
TemporalObject
  ObjectInASequence ('chordSequence' 'conceived')
    AnalysableObject ('analysisList')
      ChordInASequence ('chord' 'before' 'after')
      Shape ('subShapes')
        ChordSubstitution ('chord')
        MusicalShape ('tonality')
          AmorphousShape ()
          ComposedShape ()
            ExtendedShape ()
              ExtendedIntersectedShape ()
              GlobalShape ()
                BiTonalShape ()
                MonoTonalShape ()
                QuadriTonalShape ()
                TriTonalShape ()
          EmpruntModal ()
          KnownShape ()
            BluesShape ()
            BluesySubstitution ()
            Resolvante ()
            TurnAround ()
            TwoFive ()
            TwoFiveOne ()
        TransitionChordSequence ('beforeTonality' 'afterTonality')
```

Figure 7. The various classes representing abstract analysis objects (attributes are between parenthesis).