Musical Knowledge: what can Artificial Intelligence bring to the musician?

Geraint Wiggins* Alan Smaill*

Abstract

In this paper we introduce the idea of *Knowledge Representation* in the context of *Artificial Intelligence*. We explain why finding a good representation is an important issue in Artificial Intelligence research, and why music raises some particularly interesting questions. We then proceed to explain our own system for music knowledge representation, *Charm*.

1 Introduction

In this paper, we introduce the field of Artificial Intelligence (AI), especially as it relates to music and musicology, and discuss an aspect of fundamental importance in its use for both theory and practice: the representation of knowledge, primarily for use on computers, but not necessarily so.

In the musical area, study of *Knowledge Representation* can yield fruit for both musicians and AI scientists, with music raising some significant questions of very general import in AI, and AI techniques being very usefully applicable in many sub-fields of music.

The rest of the paper is arranged as follows. We begin with a broad characterisation of the field of AI, explaining some important distinctions and stating our position with respect to them. We proceed then to suggest that AI technology can be useful to the musician, and give examples of how this might be so. We then develop the central notion of knowledge representation, and discuss how this is of interest to musicians and workers in AI alike. This leads us in to a detailed but informal description of a particular representation system for music on computers, Charm. Finally, we draw conclusions, and suggest possible directions for future work.

*Department of Artificial Intelligence, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, Scotland. Email: A.Smaill, G.A.Wiggins@ed.ac.uk

2 What is Artificial Intelligence?

2.1 A science of intelligent behaviour

Artificial Intelligence is the scientific study and simulation of intelligent behaviour. It is sometimes incorrectly presented as a sub-field of computer science, but in reality is a strongly interdisciplinary melting-pot for ideas from many areas of human – and animal – endeavour. At an academic conference on AI, for example, one might expect to meet psychologists, biologists, linguists, logicians, mathematicians, philosophers, and other researchers interested in the scientific aspects of their fields – including, of course, musicians.

2.2 A field with many sub-fields

As a field of study, AI is divided into fairly clearly delineated sub-fields, such as: machine vision – modelling of animals' and humans' ability to understand visual images; natural language and speech processing – understanding and generating meaningful text and speech; mathematical reasoning – simulating the reasoning process of mathematicians involved in, for example, building formal proofs. One of those areas, called variously Music & AI, Cognitive Musicology and Computational Musicology, includes the subject of this paper: the representation of musical knowledge on computers.

2.3 Why study AI?

The motivation behind AI, for the majority of researchers, is somewhere on a spectrum between two complementary points of view. At one end of the scale sits Cognitive Science, whose aim is to understand the working of the mind (and also often the brain), for its own sake. At the other resides Intelligent Systems Engineering, whose purpose is to build computer systems which can perform tasks often thought of as requiring intelligence in humans, such as reacting to spoken language. The reasons for wanting to built intelligent systems vary. For example, the pervasive filtration of computer technology throughout society means that more and more people are needing to use computers without expensive training, so if the machines can understand spoken language, they are much easier for most non-programmers to use. On a more technical level, humans are good at solving certain classes of problem (for example, finding mathematical proofs) which are, in general, hard to solve by standard techniques in computer science – human mathematicians seem to use more intuitive approaches. While AI will never be able to *solve* these problems in general, its techniques are often able to go rather further than the standard "unintelligent" approaches.

2.4 How should we study intelligence?

Both the cognitive scientist and the engineer at the two extremes of AI research are involved in building systems which can then serve different purposes. In the case of

the engineer, the concern is not to match up closely with human capabilities, but to provide a system capable of performing a given task, usually inspired by some ideas of how it is done by humans. In this case, the system must be tested and revised if the performance is unsatisfactory. In the case of the cognitive scientist, the interest is in building a theory of how some human or animal cognitive activity works (for example, recognising metrical patterns in music). If such a theory is embodied in a computer system, then the performance of the system can be compared with human performance, and the theory either accepted for the moment, or, more likely, changed.

It's important to understand that this approach does not necessarily explain exactly how the mind works; we can never know if our program achieves its results *in the same way* as the system it emulates, but only that it does indeed achieve those results. However, if a model correctly *predicts* behaviour which was not explicitly programmed into it, we may draw the tentative conclusion that our model is "accurate", at least with respect to the behaviour in question. Also, often, the way in which a model fails suggests how to improve it. One particular advantage of computational modelling as a methodology is that it divorces us from the need perform gruesome experiments in the so-called "wetware" – the human brain – itself!

2.5 Styles of representation in AI

Symbolic AI

Another broad distinction we can make in the AI world is in terms of how the program embodying our simulation actually works. The distinction is between *symbolic* and *subsymbolic* systems, and it is very relevant to our discussion here. For further discussion of the difference between symbolic representations and the alternatives, see [BA91].

Symbolic AI is based on the principle of using a computational model which explicitly represents aspects of the problem being solved. In other words, the program has a given vocabulary of *symbols*, which are used to represent objects and/or concepts; it may have a model of the world in which it operates (a bit like a geographical map). In this context, the programmer knows what her chosen symbols mean in relation to the world in which the program is supposed to operate. So she can take her vocabulary of symbols, and say, in its terms, how the program achieves its results.

There are several computer languages, such as Lisp and Prolog, and even purpose-built computers, such as the Xerox Lisp Machine series, specially designed for this kind of *symbolic processing.*

Subsymbolic AI

The other general style of AI programming is the *subsymbolic* approach. The best known way to achieve this is to use *neural networks*: systems which emulate (in a fairly loose sense) the general structure and operation of the brain. A neural network consists of a number of nodes, connected together in a certain configuration. Each node is capable of reading information from those immediately connected to it, and passing on

an output to other nodes in some way dependent on those inputs. Normally, the nodes are quite simple processors, and they are usually all the same; it is their multiplicity and the connections between them that allow them to work on hard problems. For this reason, this approach to programming is sometimes called *connectionist* computing.

A neural net is programmed by "showing" it possible inputs and training it to produce the right output for each. (The details of this are beyond the scope of this paper. See [RN95] for an introduction.) The reason this is not symbolic, but subsymbolic, is that there is no *a priori* reason to be able to see any symbolic reading of the system's evolution. In other words, we cannot (usually) look inside the network and understand how it works, and what the values passing around it mean in terms of its world. We can only see its inputs and outputs – just like the human mind.

The subsymbolic approach to computing has had some spectacular successes (see [HKP91]), but also spectacular failures, mostly due to the problem that one cannot be absolutely sure that the program is really doing what one thinks it does. For the purposes of this article, however, we are situated firmly in the symbolic area. We will choose symbols to represent musical knowledge, and we will say *a priori* what they mean and how they may be reasoned with. The reader interested in the symbolic/subsymbolic dichotomy specifically in a musical context should refer to [Lem88].

3 AI for Musicians

Having broadly defined Artificial Intelligence, we now consider what use it might be to musicians, and, indeed, what use music can be to AI.

Music is undeniably an intelligent activity, which operates at more than one level in the mind. Some kinds of musical activity are deeply and consciously intellectual; some involve enormously complex and delicate physical skills, which have to be learned; and some involve what seems to be an almost subconscious and involuntary response.

Like the study of AI, the study of music, ranging from practical instrumental performance to deeply academic historical scrutiny, is a broad church, and for this reason, the two fields are able to interact very usefully at many points along the spectrum described above, between Cognitive Science and Engineering.

Beginning at the former end, we can use computational modelling to study issues of perception of music [JSM90]. We can try to build programs which emulate human performance practice and style, and thus shed light on how the player communicates her music to the listener [Des93, Des92]. We can build programs which attempt to improvise in certain styles and thus try to explain this aspect of the compositional process [BDV86, DMR87].

AI techniques can be useful to musicologists, too. New techniques for text and score analysis mean that works can be statistically analysed to help identify their composer [Cop91]. Computer modelling is helping ethnomusicologists understand the nature of non-Western musics [BK92]. New techniques for human computer interfacing are opening a potential new world of music education for everyone [SSW94, CFIZ92,

HEC90].

Much of this work makes an interesting contribution to knowledge in itself, but it is often much more than a purely theoretical activity. At the intelligent systems engineering end of our scale, these ideas can be used to analyse problems with performers and so help correct them; they can help with computer based specialist education, especially in locations or organisations where specialist human expertise is unavailable. In a performance context, computers can be allowed to interact with human performers and thus to enhance and extend the range of what is musically possible [Mac92, Row93].

Because music is such a fundamental part of human behaviour at all sorts of levels – socially, educationally, sociologically, emotionally – we may also expect its study to shed light on the working of the mind and/or of the brain. We can perhaps use music to study the relationship between intellectual and affective ability, since it seems to bridge whatever gap may exist between the two [Slo85]. We can set new challenges to AI researchers, such as the management, in real time, of the enormously complex interactions between human and computer performers. This last is a specific example of a general problem of how to schedule more or less prescribed events between two or more parties – a problem evident, for example, in today's robotic production lines.

In the area of interest of the current paper, music throws up some important general issues, which we discuss in more detail in later sections. The first of these is the question of multiple interpretations of knowledge. Musical structure is rarely definable in any single absolute sense, and even when it is so at the physical level of notes, it may not be so in an audible or psychological way.

Before we discuss these issues in detail, however, we must first introduce the general notion of *Knowledge Representation.*

4 Representing Knowledge

4.1 Why do we want to represent knowledge on computers?

Notwithstanding the existence of subsymbolic programming, the idea of and need for *Knowledge Representation* underpins any activity involving present-day computers. (In the case of neural networks, the program which manages the learning of the network is typically one that can be understood in symbolic terms at one level, but only because it deals with notions such as that of *network* and *update* of network properties; on the level of musical meaning, there is usually no use of symbols referring to the musical domain whose properties are being learned.) In one sense, even the abacus has the ability to represent a very specific kind of mathematical knowledge and to allow the user to reason with it – albeit in a very limited way. In the current paper, the kind of knowledge we are interested in is of a rather more general nature. Rather than being restricted to a particular number system, we wish to be able to make general statements about "what is true", and to draw conclusions from them.

4.2 Managing knowledge

Over the past forty years or so, computer science has advanced to a level where the activities of computer system and program design are really quite well understood. This progress has resulted in the production of so-called "high level languages", in which the programmer can separate herself from the nuts and bolts of how a program works, and think in terms of more abstract – and hence clearer – structures and processes. The nuts and bolts are then left to the automated parts of the programming system. As a result of this, the expression of program content has moved from being very close to the inner workings of the machine ("Do this, then do this, then do this…") to being much closer to an abstract definition of the problem being solved ("This is what needs to be done.").

The design of these structures and processes, and the means by which the computer finds its way around its data, are very important to the operation of a program, and to our understanding of it. The structures can also encode knowledge *about* the data as well as the data itself, and this can be very important in the operation of the program, and in updating it or correcting programmer errors. For a simple example, putting a collection of representations of objects all in one place might signify that the objects have a certain useful property in common.

The structuring and representation of knowledge in symbolic AI programs is particularly important, for several reasons. For one thing, AI programs tend to have to deal with large bodies of densely interconnected information; the data needs to be structured in such a way that a program can choose paths to move around it fairly arbitrarily. For example, the concept of a "red ball" (the coloured round thing, not the Communist Party Annual Dinner-Dance) might be approached from (at least) two directions – from the collection of "red things" or from the collection of "round things". This bifurcation applies every time we add a new piece of information to the description of an object or concept – moving from "red ball" to "big red ball", for example. See [CFIZ92] for a musical system that deals with these issues.

It is also a common feature of descriptions of objects that their properties need to be described in terms of the context in which they are found. For example, a computer may be part of the InterNet, with respect to which it has a particular, well-defined function; on the other hand, someone planning the layout of an office may well view it as no different from any other piece of furniture. The remote InterNet user does not care where the computer is; but the office planner must consider issues such as accessibility, and the location of power points. Conversely, the user of the office needs to understand more of the nature of the computer, so as to know not to splash coffee on it or to eat it or to use it to stub out cigarettes. So we need to be able to deal not only with very dense and detailed information, but also with different subsets of that information, depending on our viewpoint.

4.3 Understanding the properties of our data

On a more technical side, we would like to know something about the properties of our vocabulary of symbols itself. It would be convenient to think of it as a language,

and we would like to be sure that whatever we write down in that language can be understood. For example, there is little point in writing a program which gives us as output a bunch of symbols from our vocabulary, but whose *combined* meaning we cannot understand. In the same vein, we would like to be sure that our program will stick to the language we have given it, and not produce answers in terms of symbols whose individual meaning we have never defined.

These notions do not just relate to the language of a symbolic program, but also to the reasoning we can perform using the language. Specifically, we want our AI program to carry out some kind of *reasoning.* We certainly do not want the program to be able to draw conclusions which are incorrect or contradictory; and, often, we want to be sure that it is able potentially to draw every conclusion which is correct. These two properties are called, respectively, *soundness* and *completeness* of the reasoning system. For some languages and associated reasoning systems it is possible to prove soundness and completeness.

And we can sometimes go further than this. Given a language for knowledge representation, we can sometimes say, in a very precise mathematical sense, how difficult reasoning about knowledge expressed in a particular language is. This means that, if we are careful, we can be in a position to choose the simplest possible kind of language required to represent and reason with our knowledge. Thus, we can make the most efficient possible use of our computing facilities, and allow users to obtain their answers in the shortest possible time. See [DLNN91] for technical details.

5 Representing Musical Knowledge

5.1 Surfaces of abstraction – the musical surface

Another important question is a philosophical one: how do we choose the level of detail at which to work? This is a question of great importance in all symbolic AI. For example, in the musical area, there is a scale of abstraction along which we can move, taking us, for example, from the actual variations in air pressure which cause a sensation of sound, through notions such as “notes”, “phrases” and, eventually, “symphonies” or “concerti”. In some contexts, we might want to go even further, and consider, say, the entire output of a given composer as one large work, with a notion of controlled development as time proceeds. Further, as any music analyst will tell us, there are other dimensions along which to move which bear little or no relation to these rather Western classical terms. There is no one right answer to this question, but one needs to have *an* answer before beginning to think about working with musical, or indeed any, knowledge on a computer.

To be more specific, in order to build a representation of knowledge, we need some basic building blocks with which to start, and we would like to be consistent about them. In AI music work, the level of these blocks is sometimes called the *musical surface.* Jackendoff [Jac87] defines it as “the lowest level of representation which has musical significance.”

It is important to note that, in choosing a musical surface, one is not making a general

statement about what is or is not important for all musics. Rather, one is defining what one's own representation is intended to represent. Subsequently, one is limited to the expression of elements of the surface or constructions built from them, so it makes sense to think hard about what is the right place to start. We return to this issue in the context of our work on the Charm system, below.

5.2 How can we evaluate musical representation systems?

An example of a well-known representation which often does *not* represent what many users want is MIDI [Rot92] (the Musical Instrument Digital Interface) – its musical surface is usually quite adequate in the application of controlling electronic keyboards (for which it was designed); however, it is often used for applications well outside its range of expertise. Such users would be well advised to search the literature for a more general and expressive music representation.

MIDI has another shortcoming in terms of general representation: even if its musical surface is right for one's application, it provides very little in terms of representation for higher-level aspects of music, such as phrasing, or less audible relationships within the data. In [WMSH93], we proposed two dimensions along which musical representations may be compared. The first of these is *expressive completeness*, which measures the amount of detail which can be represented; the second is *structural generality*, which measures the amount of information about musical structure which can be encoded explicitly. MIDI scores poorly on both. It is expressively incomplete because it throws away a lot of important information (*e.g.*, timbre, attack), and it is structurally ungeneral because there is no means for the explicit representation of, say, phrasing, trills, or crescendi. All of these latter aspects of music can of course be "written out" in MIDI – small timing variations indicating phrasing, for example, or a series of alternating notes constituting a trill – but there is no notated structure which actually says "this is a phrase" or "trill this". As a result, the representation is impoverished, because musical structures like this must either be ignored or searched for again and again. We would consider a music copyist who chose without very good reason to write out trills explicitly, with no trill marking, a poor practitioner indeed.

A music representation system, then, must be flexible. Ideally, it should not restrict its user to a particular flavour of data, nor should it restrict her to representing her data in a particular syntax. A representation system such as this would be general indeed, and, unfortunately, it would be hard to make useful claims about the properties of the language, which we discussed above.

A good compromise, then, is a representation with an explicit, but not too restrictive, musical surface, within which the widest possible range of data can be represented. We show one way to achieve this below. The representation must also allow the user to make groupings of the building blocks of the musical surface, and groupings of groupings, to allow arbitrarily complex annotation of the data. In this way, a representation can maintain maximal expressive completeness (within the domain of its musical surface) and still achieve as much structural generality as is required.

5.3 Using musical representation systems

What problems, then, do knowledge representation systems for music solve?

Perhaps most obviously, they allow us to represent aspects of music which are not captured in the conventional score in well defined and generally meaningful ways – regardless of whether we want to use a computer or not. Also, they allow us to get musical knowledge into a computer file in an expressive, purpose-built way, so that we can be sure of representing precisely what we want to represent. This leads us into the world of computer aided music and musicology, as discussed above, and which needs no more advertising.

At a lower level, though, some useful practical possibilities are worth mentioning. Use of computers for music and musicology can greatly facilitate sharing of data between researchers, especially given the advent of the InterNet and the Worldwide Web. Some researchers have given careful thought to maximising the "shareability" of their knowledge representation, so that programs and data can be interchanged with the minimum of fuss.

Didactically speaking, this kind of knowledge representation opens up a significant range of possibilities for general musical data storage. By designing her representation carefully, a user can annotate a computer representation with *exactly* the information in which she is interested, in a much more flexible and more accurate way than is generally possible with, say, extensions to conventional score notation. This is because formal knowledge representation gives us a more precise way to say what our annotations mean.

Finally, back at the abstract level, using explicit, formal knowledge representations can lead us to a deeper understanding of what we mean when we think and talk about music, and a better means than hitherto of expressing and exchanging new and perhaps difficult ideas.

6 The Charm System

In this section, we describe a framework for representing music, so that it can be reasoned about, and thus the musician can be enabled to create, analyse or transform musical material. The system has been described in detail elsewhere (see [SWH93, HSW91, WMSH93, SWM93]); here we present an informal account of the principles behind the framework.

6.1 The musical surface

As was discussed in section 5.1, there is a choice to be made over where the *musical surface* is to be placed. In this case, our primary interest is in music at the level of notes and phrases, rather than the internal structure of individual tones and timbres. Therefore we base the representation around the level of musical *notes* as the lowest level of description that we work with. This does not mean that issues of timbre and intensity are ignored; it simply means that these are taken to be properties of notes,

rather than phenomena to be further analysed in some way. In choosing a note-based musical surface, we follow the choice made by Lerdahl and Jackendoff in [LJ83].

Not all sorts of music fit gracefully within a view organised around notes – in particular, much electronic music is better thought of in other ways. However, a large proportion of western music, whether tonal or atonal, can be well understood in terms of collections of notes with various properties, and thus will fit into the framework we describe.

Another choice that we have made is to represent music *in performance* rather than music *as scored*. The notion of performance is, to be sure, idealised, in that the differences from performance to performance are passed over, although still features of the score such as time signature and key signature do not feature at the primitive level of our framework. Of course, the information embodied in such notations in a score is important in understanding the significance of the music, and this information *can* appear elsewhere in the framework in another form, where it tells us about the tonal or metrical analysis of the various collections of notes in question.

6.2 Pitch, time, etc.

Two crucial properties of notes are their *pitch* and their place in *time*; the latter has two aspects, the time at which a note begins, and the time it ends (or the length of time that the note lasts). The western score notates these roughly in a two-dimensional way, where vertical displacement indicates pitch, and horizontal displacement represents time, but this has been refined to allow more accurate indication of both dimensions.

For the purpose of harnessing the power of a computer representation, the score itself is not very helpful; it is possible to take a score, considered as a pattern of marks on paper, complete with whatever smudges and stains it may contain, and turn it into a representation that would allow that pattern to be reproduced on the page, say after transmission to a different location. This is like transmitting a score by fax, since no understanding of the musical content is needed. But this is not how the musician understands the score – rather, the marks on the score refer to real or imagined musical events, and it is the structure of this music that we want to capture, not the squiggles on paper.

How can we represent the properties of a note in a more useful way for our purposes? Many ways have been used in computer programs. For example, we can measure pitch in piano music as frequency measured as a number of Hertz, or as a number of semitones above some chosen lowest note, or by giving an octave number, and a letter from A to G, and (possibly) an accidental (such as ♯), and many other ways besides. For time, there are equally many possibilities.

Rather than fix on one of the choices mentioned above, we have chosen to use what is known in Computer Science as an *abstract data type* [AHU83]. This means that we say what properties we want of the pitch and time representation, and what operations we should be able to carry out, rather than choosing a particular representation. The idea is that we don't care whether pitch is given in Hertz or semitones, and we should be able to understand the music just as well whichever choice has been made. So it

is a good idea to hide the unimportant details of that choice, and just think about the operations we need, and how they fit together.

For example, for pitches, we require operations that

- given two pitches, say what the interval is between them;
- given a pitch and an interval, say what the pitch is that interval above the given pitch;
- given two intervals, say what interval is got by adding the two intervals together.

Together with similar operations for temporal structures, this is enough to let us manipulate the pitch and time aspects of musical notes. Individual notes are associated with a particular pitch, start-time and duration.

It's important to note that the general framework encompasses a wide range of possibilities; for example, it is not restricted to semitone differences between pitches, as quarter-tone or other less standard possibilities can be dealt with just as well. Whatever particular choice is made at the end of the day, some implementation of the choice is needed, though subsequent use of the representation does not depend on this choice.

A whole piece of music is then represented as a collection of notes, each of which is situated in time and in pitch, and indeed with respect to intensity and timbre. Other properties of individual notes, such as attack, could also be included here, though our existing implementation does not support this.

6.3 Higher-level structures

Having established this note-level representation, we now discuss how musical concepts such as phrase, motif, key, inversion ... can be dealt with on top of this foundation. We particularly want to be able to deal with several simultaneous accounts of the same raw musical material; such ambiguity both in analysis and creation is a pervasive feature of the musical domain (see for example [CW95, Min81]).

Consider the brief extract shown in figure 1[1].

If we think of the notes as they would be performed, with varying intensities and timbres, as given to us, we want to be able to think of groups of the notes as fitting together for various reasons. In this case we might be interested in the perceptually dominant notes at certain moments, or in the notes with similar durations, or with notes that fit in a particular place in the twelve-note series that underlies this movement, as given in [Bou63] and shown in figure 2.

We call any such grouping a *constituent,* and it may group together not only notes, but other constituents as well. We have two ways of representing such groupings.

The first simply allows some collection of notes (or indeed some collection of structures of the sort we are in the process of describing) to be gathered together and labelled by the user as significant. For example, the notes in the first staff could be gathered

[1] not all the phrase marks and dynamic indications from the score are shown.

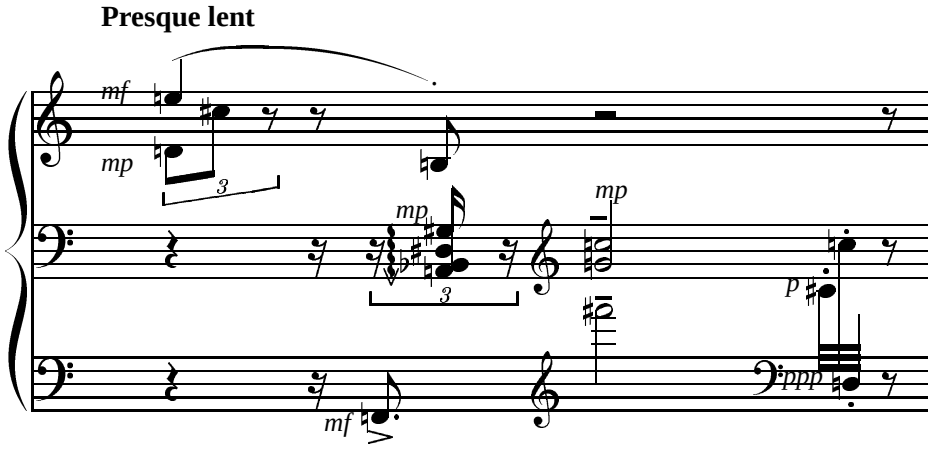

Figure 1: Boulez, third sonata, formant 2, *texte*

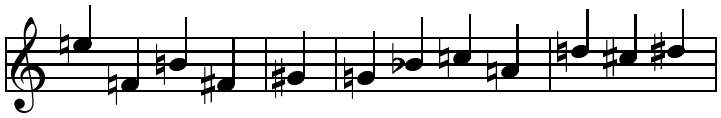

Figure 2: Series for *texte*

together as follows. (Although the staff layout is not significant at the note level of the representation, we may suppose the composer associated some structural information with this layout in this case that an analyst might like to explore.)

Suppose that these notes are associated with statements of "events", each given a name `e1,e2,e3` ..., and specifying information about the pitch, time, intensity *etc* characteristics (we omit the details of these here; they should be chosen according to the discussion in the previous section).

```
event( e1, pitch( ... ), time( ... ), ... ).
event( e2, pitch( ... ), time( ... ), ... ).
event( e3, pitch( ... ), time( ... ), ... ).
event( e4, pitch( ... ), time( ... ), ... ).
event( e5, pitch( ... ), time( ... ), ... ).
.
.
```

Let's suppose that the notes in the top staff correspond to `e1,e2,e3,e5`. Then a statement can be made that these events together form a collection we call a *constituent*, of a type we'll call a *motif* in this case. This is written as

```
constituent( c1, motif(1), { e1,e2,e3,e5 }, ... ).
```

indicating that the constituent labeled `c1` is an instance of `motif(1)`, containing the events `e1,e2,e3,e5`.

For this sort of constituent, any collection is possible, and the labels and classes assigned are arbitrary. There is another sort of constituent where, instead of saying what things are to be gathered together, a logical specification is given of what properties the members of the collection should have in order to be gathered together in this way. For example, a collection of notes will form a *series* just if their pitches correspond one to one with the twelve semitones of an octave, in some order, and ignoring the octave of the pitch. We can write down in our specification language this characterisation; this means that we can have the computer analyse such music to look for collections of notes which form a series, or which check collections of notes to see if they form a series.

For another example, consider the motif we defined above. What is important for a motif is to be able to recognise other occurrences of the motif, which will differ typically in time, pitch, and other characteristics. However, in a particular style we expect to be able to characterise this relationship of similarity, for example, by saying what sorts of transpositions, augmentations *etc* can be allowed between motifs. Again this allows the system to do several tasks – it can check an analyst's work when she identifies several collection of notes as being instances of the same motif; it can search through music in order to find instances of a given motif; it can search for two collections of notes which are close enough to count as a motif (this is the basis of the work of [Cop91]); and it can be used to generate variants of a given motif.

In the case of the example in figure 1, Boulez in [Bou63] quotes the series shown in figure 2. The series is divided into motifs as indicated. We can define similarity of motifs in this context by saying that we are looking for any transposition together with possibly an inversion that will make the pitches agree (ignoring octaves, and ignoring the order of notes). Now we can test to see if the motif chosen is the same as one of the motifs in the series – it is, and in more than one way, given the symmetries in this series.

So far we have seen examples of how notes can be gathered into collections. Further collections, which can pull together collections already defined, allow us to represent other features. For example, the arpeggiated chord in the example above can also be analysed as corresponding to a motif from the series. We can then ask if the two motifs from the example come from a common transformation of the series (which is the case), and then of the rest of the series can be found in the extract shown. Further, the enchaining of versions of the series which forms the basis for this movement can equally be detected in this manner, and represented as a collection of (overlapping) constituents.

This example lends itself more to logical analysis than music written in other styles. Nevertheless, we believe that a wide range of styles can be treated in this way, and that the logical relationships that underly musical analysis can be made explicit. Indeed, one of the advantages of such an approach is that it forces the analyst to be explicit about her underlying assumptions.

We should also point out here that it is possible to provide assistance in the task of

working out general rules, where some set of specific examples of analysis is known, but where the general characterisation is not immediately available. This area is known as Machine Learning; see [SWM93] for an overview.

In the space available here, we can only hint at the role we envisage for our framework within composition, and within transformation of musical material, for example in a situation where musicians are interacting in performance with a computerised system. Our claim is that what is needed is a language for dealing with musical material that is both accessible to the composer and performer, and which is interpretable by machine; this view is also taken by builders of substantial musical computer systems, such as that described in [CFIZ92]. For our framework to be useful in this role, a substantial library of logical characterisations for the the central musical concepts is needed; this is the direction that future work will take.

7 Conclusion

In this paper we have informally outlined the basis of a generalised music representation for use with computers. We placed this in a context of an introduction to the ideas of AI, and a discussion of some of the possible uses of AI technology in the musical world.

The subject matter of this paper is broad indeed, and therefore we can only begin to suggest the breadth of interesting possibilities in such a short space. We have attempt to emphasise, however, that the benefits of the meeting between music and AI are manifold and of interest to both parties, on both practical and theoretical levels.

We stand at the doorway to an exciting new world of musical experience, activity and understanding, which can, in the longer term, only benefit the understanding of humankind and its currently inexplicable urge to make and hear music. The concepts and principles presented here are absolutely fundamental to this marriage of science and art; having laid them out in theory, we will now continue to apply them in more applied or practical directions.

References

[AHU83] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Data Structures and Algorithms.* Addison Wesley, Reading, Massachussetts, 1983.

[BA91] W. Bechtel and A. Abrahamsen. *Connectionism and the Mind.* Blackwell, 1991.

[BDV86] W. Buxton, R. Dannenberg, and B. Vercoe. The computer as accompanist. In *CHI '86 Conference Proceedings,* pages 41–43. ACM/SIGCHI, April 1986.

[BK92] B. Bel and J. Kippen. Bol processor grammars. In O. Laske, M. Balaban, and K. Ebcioglu, editors, *Understanding Music with AI – Perspectives on Music Cognition,* pages 366–401. MIT Press, Cambridge, MA, 1992.

[Bou63] P. Boulez. *Penser la musique aujourd'hui.* Gonthier, Mayence, 1963.

[CFIZ92] A. Camurri, M. Frixione, C. Innocenti, and R. Zaccaria. A model of representation and communication of music and multimedia knowledge. In B. Neumann, editor, *Tenth European Conference on Artificial Intelligence*, pages 164–8, Vienna, 1992. John Wiley and Sons, Chichester, England.

[Cop91] D. Cope. *Computers and Musical Style.* Oxford University Press, 1991.

[CW95] D. Conklin and I.H. Witten. Multiple viewpoint systems for music prediction. *J New Music Research*, 24(1):51–73, March 1995.

[Des92] P. Desain. A (de)composable theory of rhythm perception. *Music Perception*, 9(4):439–454, 1992.

[Des93] P. Desain. A connectionist and a traditional AI quantizer, symbolic versus sub-symbolic models of rhythm perception. *Contemporary Music Review*, 9:239–254, 1993.

[DLNN91] F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proceedings of IJCAI-91*, volume 1, pages 458–463. International Joint Conference on Artificial Intelligence, 1991.

[DMR87] R. Dannenberg and B. Mont-Reynaud. Following an improvisation in real time. In *Proceedings of the International Computer Music Conference*, pages 241–248. Computer Music Association, August 1987.

[HEC90] S. Holland and M. Elsom-Cook. Architecture of a knowledge-based music tutor. In M. Elsom-Cook, editor, *Guided Discovery Tutoring*, London, 1990. Paul Chapman Publishing Ltd.

[HKP91] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Company, Redwood City, CA, 1991.

[HSW91] M. Harris, A. Smaill, and G. Wiggins. Representing music symbolically. In *IX Colloquio di Informatica Musicale*, Genoa, Italy, 1991. Also available from Edinburgh as DAI Research Paper No. 562.

[Jac87] R. Jackendoff. *Consciousness and the computational mind.* MIT Press, Cambridge, MA., 1987.

[JSM90] Jacqueline A. Jones, Don L. Scarborough, and Benjamin O. Miller. Gtsim – a computer simulation of music perception. In *Colloque International - Musique et Assistance Informatique*, Marseille, 1990. From the proceedings of the Marseille conference.

[Lem88] M. Leman. Symbolic and subsymbolic information processing in models of musical communication and cognition. *Interface*, 18:141–60, 1988.

[LJ83] F. Lerdahl and R.S. Jackendoff. *A Generative Theory of Tonal Music*. The MIT Press, Cambridge, MA., 1983.

[Mac92] T. Machover. Hyperinstruments : A progress report 1987-1991. Media lab document, MIT, Cambridge, MA, 1992.

[Min81] M. Minsky. Music, mind and meaning. *Computer Music Journal*, 5(3):28–44, 1981.

[RN95] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall International, London, 1995.

[Rot92] J. Rothstein. *MIDI : a comprehensive introduction.* Oxford University Press, Oxford, 1992.

[Row93] R. Rowe. *Interactive Music Systems.* MIT Press, Cambridge, Massachetts, 1993.

[Slo85] J. Sloboda. *The Musical Mind.* Oxford Science Press, Oxford, 1985.

[SSW94] M. Smith, A. Smaill, and G. Wiggins, editors. *Music Education: An Artificial Intelligence Perspective, Edinburgh 1993*, London, 1994. Springer Verlag. Workshops in Computing series.

[SWH93] A. Smaill, G. Wiggins, and M. Harris. Hierarchical music representation for analysis and composition. *Computers and the Humanities*, 27:7–17, 1993. Also available from Edinburgh as DAI Research Paper No. 511.

[SWM93] A. Smaill, G. Wiggins, and E. Miranda. Music representation – between the musician and the computer. In M. Smith, G. Wiggins, and A. Smaill, editors, *Music Education: An Artificial Intelligence Perspective.* Springer Verlag, London, 1993. Also available as DAI Research Paper 668.

[WMSH93] G. Wiggins, E. Miranda, A. Smaill, and M. Harris. A framework for the evaluation of music representation systems. *Computer Music Journal*, 17(3):31–42, 1993. Machine Tongues series, number XVII; Also available from Edinburgh as DAI Research Paper No. 658.