The Use of Constraint Systems for Musical Composition

Geraint A. Wiggins
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland

Abstract

I discuss three issues relevant to the use of constraints in musical AI applications: four-part harmony; serial composition; and music representation. I describe two small pieces of work in the former two, and suggest that the last constitutes an important challenge question for those interested in designing constraint systems for use in an AI/Music context.

1 Introduction

In this paper, I discuss three issues which are relevant to the use of constraint programming (specifically in my case, constraint *logic* programming) in musical AI applications. The paper is not intended to be a report on work carried out (though it does contain some such) – rather, I aim primarily to raise some important issues which seem to be largely ignored by the research current in the field.

First, in Section 2, I outline some experiments carried out using a Genetic Algorithm for four-part harmonisation, which sheds some light on exactly why the problem is hard for any unstructured search based approach. The same reasoning suggests to some extent that constraint methods are likely to be relatively successful.

Second, in Section 3, I describe how I used a simple CLP(FD) based system to construct musical material on which I recently based a composition.

Third, in Section 4, I discuss some subtleties of the common notation for pitch, and some common assumptions which I believe are not in fact tenable in the context of studying human musical behaviour.

Finally, I conclude with a short summary.

2 Problems using Constraints to Harmonise

2.1 Introduction

In this section, I outline a recent project studying the application of Genetic Algorithms (GAs) to music. The research was carried out primarily by Somnuk Phon-Amnuaisuk, a Doctoral student at Edinburgh, and is reported in more detail in Wiggins et al. (1998) and Phon-Amnuaisuk et al. (1998).

Why am I discussing a GA system here, in a paper about constraints? Because, from one point of view, a GA is very much a constraint satisfaction mechanism. For the uninitiated, there follows a brief description of GA function.

2.2 The Outline of a GA

The basic model of a GA, derived from the Theory of Natural Selection of Darwin (1859), is that we begin with a pool of *chromosomes* each of which represents (explicitly or implicitly) a potential solution to the problem we are trying to solve – in this case, homophonic harmonisation of a monophonic melody. We supply operators which are able to *mutate* the members of that pool, and *crossover* operators which are able to combine different parts of existing chromosomes to create new ones. These two classes of operator are directly inspired by asexual and sexual reproduction, respectively, in Nature.

We also supply a *fitness function* which yields a numerical, possibly multi-objective, measure of how good a solution any given chromosome is. The fitness function is then used in one of a number of possible *selection schemes* to simulate "survival of the fittest".

GAs are a stochastic search method, and they are generally regarded as a "weak" method in that they are not problem-specific. However, we argue in Phon-Amnuaisuk et al. (1998) that using problem-specific mutations makes a huge increase in the effectiveness of GA search. Even in spite of the obvious problems with stochastic methods (*viz.* chancing to miss an optimal solutions even where one is readily available), GAs have been very successful in difficult problems such as timetabling and scheduling.

For a fuller description of Genetic Algorithm theory, see Holland (1992).

2.3 GAs as Constraint Solution Mechanisms

GAs may be viewed as constraint solution mechanisms in the following simple way: each chromosome is a potential solution which conforms to constraints specified in the fitness function to a greater or lesser extent. Solutions which fail to conform die out of the chromosome pool, so search is restricted as time proceeds to sets of potential solutions which more and more conform to the constraints in the fitness function.

In some GAs – specifically in that of our work – *directed* mutations are used to apply musically meaningful mutations to chromosomes. Directed mutations encode knowledge about the problem domain, and so cause changes which are known to be beneficial, which helps to improve the chromosomes' conformity to the required constraints.

2.4 Case Study: Four-Part Harmonisation

In the work under discussion here, a straightforward approach was taken to the problem of four-part vocal harmonisation. Each chromosome was a $4 \times n$ matrix of notes, the first row being the (given) melody, which is n notes long. The population was initialised randomly. Directed mutations applied musically sensible operations to the chromosomes, and the fitness function encoded some basic rules of harmony, which are, given the interconnected nature of chords in harmonic sequences, very much constraints.

It quickly became clear that the system was capable of doing quite well on the work that it produced – one of its harmonisations was graded a pass on the same criteria as a 1st year Music undergraduate, in an independent evaluation. However, it also became clear that the search space for solutions to most harmonisation problems is enormously convoluted. This idea is captured effectively in Figure 1.

The figure shows how the fitness value varies along the chromosome, and how this variation changes with time. High values are bad, so the high, jagged lines at the back of the graph correspond with the random initialisation, while the lower, smoother parts at the front correspond with later, more evolved generations. Note that the fitness profile does not flatten out completely: there are isolated

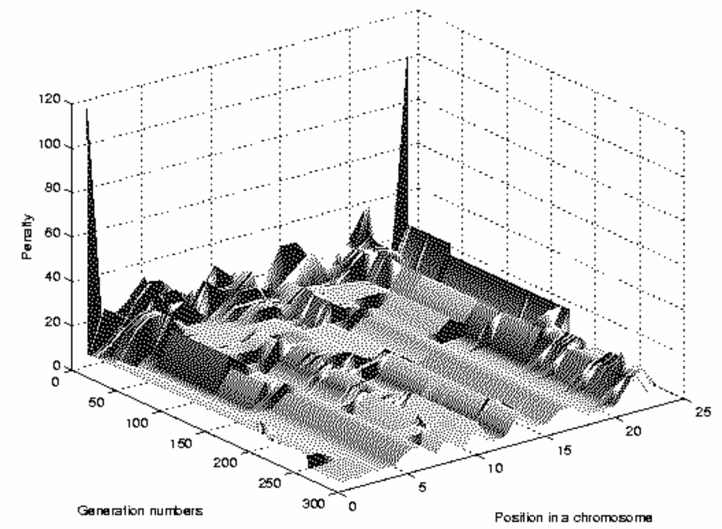

Figure 1: Fitness profile for harmonised chromosome, over time

spikes of "badness". It turns out that no amount of running of the GA will remove those spikes. Even the drastic (and unacceptably *ad hoc*) measure of supplying a highly directed mutation to flatten them out does not lead to a better solution. Details of the project, including musical output, are given in Phon-Amnuaisuk et al. (1998).

2.5 Discussion

The reason for this problem can be easily seen on examination of the search space. In the four-part harmonisation problem, where one is working on the level of individual voices, rather than on chords, it is rarely possible to change a note without having a knock-on effect on at least some, if not many, of the notes around. That is, the local quality of a given harmonic movement is *extremely* sensitive to its context.

To put this another way: the search space for this problem is very sharply convoluted, with many deep basins of attraction. Jumping from one such basin to another may involve changing many details of a chromosome at once. Standard GAs are not good at doing this, and even if they were, the chances of all the necessary mutations being randomly applied at once are very small.

The same problem will obtain not just in the GA context, but wherever search, systematic or otherwise, is used on the harmonisation problem. The use of constraint solution algorithms, therefore, while being a big help in the solution of the problem, are not a complete solution in themselves: more tradtional heuristic methods will run into just the same problems as the GA-style search described here.

I suggest that, to find a real solution to the harmonisation problem, we should look to the methods evolved by composers over the 1000 years or so during which tonal harmonisation has developed. That method uses a high-level control structure, which guides the search by making particularly difficult choices first (*e.g.,* choosing cadences, then bass lines, then inner parts). This is a prime example of a real-world least commitment strategy, and is worthy of study in an AI context in itself. In particular, composers *almost never* work linearly from start to end of the piece; neither do they expect to fill in the full vertical detail of a harmonisation at first pass.

There are several attempts in the literature to solve this problem, perhaps most notably that of Ebcioglu (1988), but there seems not yet to have been a concerted effort to make explicit the reasoning involved.

If we wish to solve the harmonisation problem, we will need to enhance the very promising constraint-based approaches with explicit meta-level reasoning controlling search to render this enormous solution space tractable.

3 Using Constraints for Serial Composition

3.1 Introduction

I now describe some work using constraints for a much more tractable problem: the formation of twelve-tone sequences in order to build compositional material. Doing so requires a little background historical knowledge, and this is where I begin.

3.2 Serial Composition

Serial Composition was introduced as a formal concept by Schoenberg (1984). It was introduced in response to a historical trend taking place over an extended period ending in the first quarter of the twentieth century. Essentially, beginning from the Classical period (*e.g.,* Mozart, Haydn), when music was generally very closely constrained by the notion of fixed tonality, there was a gradual extension of composers' musical language to include more and more pitches which were not closely related to the key in which a piece was written. Through the work of Beethoven and into the Romantic period, finally culminating in richly chromatic work such as that of Richard Strauß and Wagner, more and more notes were added into the composers palette beyond the basic seven of the major and minor scales predominantly used by the Classical composers.

Schoenberg, himself a richly chromatic Romantic composer in his early years, decided to formalise this intense chromaticism, by explicitly rejecting the idea of a tonal centre (or key) altogether. His approach to doing so was to compose using a method involving each of the twelve notes of the chromatic scale in strictly equal numbers, by building a *series* – a twelve-note chromatic sequence in which no note is repeated. For this reason, serial composition is sometimes (indeed, by Schoenberg himself) referred to as the "twelve-note method" of composition.

Schoenberg argued that one could motivate the serial approach by the natural extension of the means by which the diatonic major scale is constructed. The major scale, and, indeed, tonal harmony, is strongly motivated by physics. In particular, the major triad, the defining basis of Western tonal harmony, is derived from simple harmonic frequency ratios. Consider the A major triad shown in Figure 2. It is called a triad because it contains three notes. The frequency of the lowest note, the

Figure 2: An A major triad

A, is 440Hz[1]. This is perceived, because of its frequency relationship with the other notes, as the fundamental note of the chord – the *tonic* – and so the chord is named after it. The tonic is thought of in tonal harmony as the most *stable* note in a key – it is the note which is happiest to stay where it is, and which is least likely to imply a subsequent movement in any harmonic direction. The highest note, E, is (in a correctly tuned instrument) exactly 1.5 times the frequency of A – 660Hz. This value arises from the *harmonic ratio* of 3 to 2: exactly the frequency ratio between the second and third harmonics of a pitched sound. So E, the *dominant* note – *i.e.,* the next most important and stable after the tonic – coincides in frequency (modulo octaves) with the third harmonic of A – the lowest harmonic in the series which is not perceived as an A itself.

The C♯, on the other hand, is a less (but still fairly) stable note in the key of A major, and its frequency multiple with respect to A is $\frac{5}{4}$ – that is, 550Hz. So $\frac{3}{2}$ gives us the fifth, E, $\frac{5}{4}$ gives us the third, C♯, and so on: $\frac{9}{8}$ gives us the second (an interval of a tone above the tonic). As we use ratios which get closer to unity, we generate notes which are progressively less stable with respect to the key of A major – that is, they are progressively less likely to appear in a composition in that key, and if they do, it will probably be as part of a movement between two more stable notes. For more and deeper discussion of these issues, see Campbell and Greated (1988).

The point of all this is to demonstrate that there is a method for constructing the notes of the diatonic scale according to firm mathematical and physical principles[2]. Schoenberg justified his approach of using the chromatic, or twelve-note scale, rather than the conventional diatonic one, which gives dominance to the lower harmonic ratios, on the basis that it was simply an extension of that system to the higher harmonic ratios. An example of this is the generation of the semitone interval above the tonic, from the frequency ratio $\frac{17}{16}$.

This raises the question of consonance and dissonance – one of the features of the closer-to-unity ratios is that they give a perceptual effect which many people regard as unpleasant – a "clash" or *dischord*. In response to this, Schoenberg (1974) simply stated that "dissonance is an outmoded concept" – which may well account for the lack of public support for much serial music. That discussion, however, is outwith the scope of the present paper.

To return to serial composition itself: Schoenberg proposed that a composer might work by using the twelve notes of the chromatic scale arranged in a certain order, selected before beginning work on the piece. The properties of the series then determine some of the compositional possibilities. Figure 3 gives an example of a series from Boulez (1963). Having selected a series from the 12!

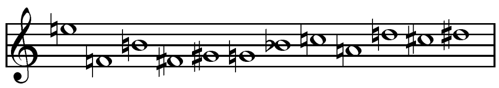

Figure 3: Series for Boulez's third sonata

available, Schoenberg suggests that four basic operations can be applied: inversion, retrograde, and retrograde inversion. Inversion means reflecting a series *horizontally* around a given pitch; retrograde means reflecting a series *vertically* in time, so it runs backwards. Figure 4 shows the inversion of the series in Figure 3 around B♮ and the retrograde form; the retrograde inversion is left to the reader's imagination. Note the properties of these transformations: they are both self-inverse; and if two successive inversions around different pitches are applied, doing so is equivalent to a transposition

[1] At least, this is the UK standard.

[2] In fact, I have glossed over the rather fundamental question of whether this theory corresponds with actual practice. In Section 4, I will suggest that things are not really this simple.

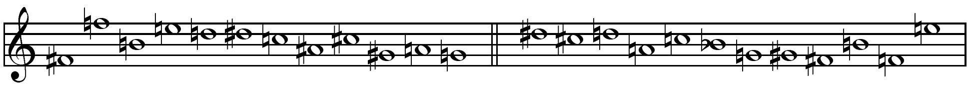

Figure 4: Inversion around B♮ and retrograde of Figure 3 series

(*i.e.,* a pitch shift of each note in the series by a fixed amount). Of course, the results of applying the transformations are guaranteed to be well-formed series themselves.

Composers, since and including Schoenberg, have used series in many different ways. For example, there is no requirement that the series be played out linearly. The Boulez series above could easily be folded into a sequence of chords, as shown in Figure 5, this particular configuration being intensely dissonant. Note that this rewriting has necessitated notational changes, since twelve-note composition goes rather against the grain of traditional (diatonic) notation. For readability, I have omitted the explicit ♮ signs.

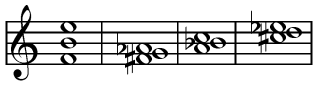

Figure 5: Chords from the series of Figure 3

Given this kind of usage, where consecutive notes of the series are being routinely juxtaposed, it follows that the form of the basic series will affect the timbral and harmonic quality of the piece. It is the freedom to manipulate the series which gives serial composers their characteristic musical languages; it is the use of the serial method which gives them their discipline.

3.3 Case Study: Generating a Series

3.3.1 Task Description

In the case study presented here, I was interested in generating material for a new work for flute, oboe, 'cello and harp, called "Elements". I wanted a short opening movement, composed serially, which had a particular quality: that it should end with a chorale (a slow chordal sequence) based around my favourite chord. There is no proper name for this chord in traditional harmony, though it is not uncommon in blues and jazz. I think of it in terms of the polyphonic harmony of the Renaissance, in calling it a "false relation chord", the reasons for this being beyond the scope of the present paper. The chord is shown, based on A, in Figure 6. Note in particular that the chord includes both the major and the minor third, and that the minor third is an octave higher than the major – a different arrangement produces quite a different effect. So I wanted to generate a twelve-tone series, which, when played

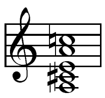

Figure 6: My favourite chord, based on A

by the four instruments, would form a sequence of twelve of my favourite chords, though relaxing the constraint about which of the thirds must be higher. I decided to assign the series to each instrument as follows:

Flute: the original series, played linearly;

Oboe: the retrograde series;

'Cello: the original series, rotated four places;

Harp: the retrograde series, rotated eight places.

By "rotated N places", here, I mean that the first N notes of the series were removed, and replaced at the end. Rotation produces the same effect as that used in folk tunes performed as rounds, such as "London's Burning" and "Frère Jacques".

Finally, I decided that I would like my series to begin with the note G, which I arbitrarily supposed to be in the middle of the range of semitones to be considered.

As to the question "Why did I make these particular choices?", the (partial) answer is this: I wanted a fair degree of symmetry in my series, but I also wanted the freedom to experiment with different structures derived from the series. I decided that one way to achieve such a symmetry would be to constrain the original and retrograde forms together, but to do so in a lop-sided way, so that the symmetry was not trivial. The lop-sidedness is achieved by rotating the different occurrences of the series with respect to each other.

Now, if we assume that all the twelve notes of our series are within the same octave, so we can only choose a given one of each pitch, there are, as stated above, 12! possible series to choose from. This is not a feasible search space for human reasoning; nor, I soon discovered, is it feasible for an approach based on exhaustive generation and testing. It is, however, very amenable to search in the style of Constraint Logic Programming.

3.3.2 Approach

It will come as no surprise to those familiar with Constraint Logic Programming over Finite Integer Domains that a very natural way to represent a series is with a list of integer variables constrained to lie between 0 and 11. In SICStus Prolog 3.5 (Swedish Institute of Computer Science (1998)), my programming language of choice, creating such a series is trivial[3]:

```
is_series( Series ) :-
        length( Series, 12 ),     % Series is a list 12 long
        domain( Series, 0, 11 ), % each element is between 0 and 11
        all_different( Series ). % no element is repeated
```

The predicates used here are all defined in SICStus libraries.

The retrograde operation can then be stated as a relation between two lists, one of which is the reversal of the other:

```
retrograde( Forwards, Backwards ) :-
        reverse( Forwards, Backwards ). % Reverse the series
```

and rotation is scarcely more complicated:

[3]Nevertheless, for completeness, I include the code, here, breaking my own general rule of not including code in research publications. I believe the code really is transparent enough to serve directly as a statement of the algorithm.

```
rotate( N, Initial, Rotated ) :-
        length( N, Moved ),             % The moved part is N long
        append( Moved, Rest, Initial ), % Take moved part off the front
        append( Rest, Moved, Rotated ). % and stick it on the end
```

Finally, we have to deal with the chording constraint. This is more complicated, and requires the use of more advanced constraint handling. Clearly, the simplest way to relate the chords together was to recurse down the four lists representing the different version of the series in parallel. But that raises a problem: in order to decide if the correct chord type is being placed at a particular position, one must be able to sort the notes into order of pitch. One cannot do this if the notes are uninstantiated. The solution I have chosen is to use the SICStus builtin `element/3`, a constraint equivalent of the more familiar `nth/3` predicate. This is just one solution to this problem, and an expert on SICStus Prolog might well improve upon it.

```
chords( [], [], [], [] ).
chords( [A|As], [B|Bs], [C|Cs], [D|Ds] ) :-
        domain( [P,Q,R,S], 0, 11 ), % make a store for the sorted list
        ( P + 3 ) mod 12 #= Q ,     % apply intervals needed to create
        ( Q + 1 ) mod 12 #= R,      % the chord, mod 12 to prevent
        ( R + 3 ) mod 12 #= S,      % overshooting the octave
        all_different( [E,F,G,H] ), % each note must be picked once
        element( E, [A,B,C,D], P ), % pick notes until they are sorted
        element( F, [A,B,C,D], Q ),
        element( G, [A,B,C,D], R ),
        element( H, [A,B,C,D], S ),
        chords( As, Bs, Cs, Ds ).   % and recurse along the series
```

Given these definitions, my series can be constrained with the following simple code:

```
series( Fl ) :-
        is_row( Fl ),               % Fl is Flute part
        retrograde( Fl, Ob ),       % Ob is oboe part
        rotate( 4, Fl, Vc ),        % Vc is 'cello part
        rotate( 8, Ob, Hp ),        % Hp is harp part
        chords( Fl, Ob, Vc, Hp ).
```

Subsequently, the variables can be enumerated via some appropriate mechanism.

3.3.3 The Series

Finally, I can now generate my series. It turns out that there are many solutions to the constraint system shown above. I chose a particular one, shown in Figure 7, because it has even more useful properties than my program required. It will harmonise with itself, yielding my favourite chord, in more ways than the one required by the constraints; and it can be thought of as being composed of two interleaved whole-tone scales, which I decided to use in the introduction to the piece.

Figure 7: Symmetrical series constrained around "false-relation" chord

3.4 Discussion

This section has presented a very empirical study of how constraint (logic) programming can be useful in music composition. I have tried to show, in this rather non-scientific style, how natural it is for a composer to think in terms of constraints, and also how natural integer finite domains are as a method of representing twelve-note music. In Wiggins et al. (1989), Smaill et al. (1993), and Harris et al. (1991) we give a more technical discussion of the issues involved in representing music, and give a more expressive alternative to the integer domain. I will return to this in Section 4, below.

4 Music Representation and Variable Temperament

4.1 Introduction

In this section, I discuss an issue which is often forgotten in the consideration of pitch systems and the expression of constraints within them. This issue concerns the correspondence between human pitch perception and the mathematical systems used to represent pitch.

I do not propose a particular solution here, though I do argue that the two obvious solutions are inadequate.

4.2 Just *vs.* Equal Temperament

During the 17th Century, a major realisation dawned upon the makers of musical keyboard instruments: that it was possible to tune an instrument in such a way that one could play in any key one liked. This, perhaps, comes as a surprise to modern musicians, who are used to doing just that all the time. At the time, the idea was so revolutionary that J. S. Bach himself wrote a set of Preludes and Fugues to celebrate the new idea: "The Well-Tempered Clavier" (Bach (1951)).

The issue is this. Until the invention of equal (or well) temperament, all the notes of the diatonic scale were tuned differently, depending on which key they were in. Even more confusing than this was the fact that notes which are now considered equivalent (*e.g.,* D♯ and E♭) were subtly different. The reason for this was that the frequencies of the notes were determined directly by the harmonic-related system outlined in Section 3.2. So the intervals shown in Figure 8 are potentially problematic. In the

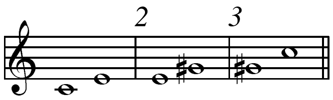

Figure 8: Three major thirds, making an octave

key of C major, the first major third, from C to E, is *different* from the second, from E to G♯, and again from the third, from G♯ to C′. If we juxtapose three major thirds in the same direction, we would want to get an octave – that is, a frequency ratio of 2. However, if we use the same third each time, say the first one, whose frequency ratio is $\frac{5}{4}$, we end up with a frequency ratio of $\frac{5}{4} \times \frac{5}{4} \times \frac{5}{4}$ which is approximately 1.953, which is not the same as 2.

The introduction of equal temperament, in which every semitone is tuned equally, being a frequency ratio of $\sqrt[12]{2} \simeq 1.059$, gave a partial solution to this problem. In this system, three major thirds is simply twelve semitones, so the ratios work out and give the desired value of 2.

One might assume, therefore, that there is no problem. However, the fact remains that many acoustic instruments, particularly those based on the harmonic series of a tube or other physical structure, do not use equal temperament tuning. The problem is even more severe for singers, because the ear tends to adapt to whatever tonal context is being imagined, and so, if the perceived tonal context is incorrect, or indeed not there at all, as in serial music, intervals can be pitched incorrectly.

4.3 Discussion

This, then, is an important issue for systems which aim to represent music, particularly if they claim to do so in a psychologically plausible way. It should be possible to represent any pitch system – in particular, systems where the precise values of notes may change, as the key of a piece modulates, for example, or where a composer has written poly- or atonally. If we do not do so, then we will not able to emulate human musical behaviour with any precision.

In Wiggins et al. (1989), we propose a system of representation for music (considered, as a starting point, as a sequence of events of constant pitch), which is based on the notion of the abstract data type. The precise details are not relevant to my argument here – the important point is that the mathematical system required for pitch (and the corresponding notion of interval) is a linearly ordered commutative group. However, to be a valid representation of the delicacy of the human ear, the operations defining any concretisation of our abstract datatype must go beyond the standard MIDI file approach of labelling the notes of a piano keyboard. We must build in the notion of tonal centre, and then express pitch with relation to that.

So we need a specialised constraint system which will work over not just the equal-tempered scale. There is a well-tried and very effective system for doing a significant proportion of this work: the familiar tonal representation of octaves, major thirds, fifths, and so on. This must be extended with other intervals which would allow us to express the difference between, say, D$\sharp$ and E$\flat$ in the same key, or, indeed, in different keys.

An obvious candidate, at least for the starting point for such a representation, is CLP over rational numbers, which will certainly allow us to express the harmonic ratios we want in intervals, but there is still the question of representing the individual notes.

Having ruled out integers and rationals, I do not propose an answer to this question here – rather I wish to raise the issue as a warning to those who would wish, as do I, to use constraint systems to generate musical material. This question of pitch representation is not as simple as some manufacturers of electronic keyboards would have us believe.

It seems that this issue has not been tackled in music representation research so far, at least as far as the survey of Wiggins et al. (1993) could determine. If we are to represent the behaviour of expert musicians effectively, then this is a significant challenge problem which must be addressed.

5 Conclusion

In this paper, I have discussed three issues of interest to those who would use constraint technology in the creation of music and the simulation of human musical behaviour.

First, I addressed the problem of four-part harmonisation, an old favourite, and pointed out that the search space is not amenable to unstructured searching, even with the power of constraint technology, due to its intensely convoluted nature. I suggested that looking at the methods used by human composers it probably a good way to proceed – in particular, it is important, if we are to elucidate the process that control reasoning be *explicit*, rather than implicit, or at least encoded, as in some existing systems.

Second, I outlined a simple system designed to help me in the composition of equal-tempered, twelve-note composition, and suggested that Constraint Programming with Integer Finite Domains is a very good solution.

Thirdly, I went on to point out that for the majority of music – that is to say, anything tonal – the approximation afforded by the equal-termpered scale may not be adequate, and that if we wish to explain and/or emulate human musical behaviour, we need a better representation, which accounts for the harmonic physical basis of human pitch perception.

References

Bach, J. S. (1951). *The Well-Tempered Klavier*. Associated Board of the Royal Schools of Music, London.

Boulez, P. (1963). *Penser la musique aujourd'hui*. Gonthier, Mayence.

Campbell, M. and Greated, C. (1988). *The Musician's Guide to Acoustics*. Schirmer, New York.

Darwin, C. (1859). *On the Origin of Species*. John Murray, London.

Ebcioglu, K. (1988). An expert system for harmonizing four-part chorales. *Computer Music Journal*, 12.

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

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*. Cambridge: Massachusetts: M.I.T. Press.

Phon-Amnuaisuk, S., Tuson, A., and Wiggins, G. A. (1998). Evolving musical harmonisation. Research paper, Department of Artificial Intelligence, University of Edinburgh.

Schoenberg, A. (1974). *Letters*. Faber, London. Edited by Erwin Stein. Translated from the original German by Eithne Wilkins and Ernst Kaiser.

Schoenberg, A. (1984). *Style and Idea: selected writings of Arnold Schoenberg*. University of California Press, Berkeley. Edited by Leonard Stein. Translated from the original German by Leo Black.

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

Swedish Institute of Computer Science (1998). The SICStus Prolog user manual.

Wiggins, G., Harris, M., and Smaill, A. (1989). Representing music for analysis and composition. In Balaban, M., Ebcioglu, K., Laske, O., Lischka, C., and Sorisio, L., editors, *Proceedings of the 2nd IJCAI AI/Music Workshop*, pages 63–71, Detroit, Michigan. Also from Edinburgh as DAI Research Paper No. 504.

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

Wiggins, G. A., Papdopoulos, G., Phon-Amnuaisuk, S., and Tuson, A. (1998). Evolutionary methods for musical composition. In Naranjo, M. and Deliège, I., editors, *Proceedings of the CASYS'98 Workshop on Anticipation, Cognition and Music*, Liège, Belgium. Also available as a Research Paper from the Department of Artificial Intelligence, University of Edinburgh.