

LOGICAL REPRESENTATION OF MUSICAL CONCEPTS

(for Analysis and Composition Tasks Using Computers)

Somnuk Phon-Amnuaisuk

email: somnuk.amnuaisuk@mmu.edu.my

Music Informatics Research Group,
Faculty of Information Technology, Multimedia University,
Jln Multimedia, 63100 Cyberjaya,
Selangor Darul Ehsan, Malaysia

ABSTRACT

The main activities in music studies concern defining structures and studying relationships among them. Computers could be very useful in performing these tasks. This paper discusses the representation of musical concepts for reasoning tasks which are common in the analysis and composition of musical works. We argue in favour of the following properties in our representation framework: *abstraction, expressiveness, multiple views* and *reasoning efficiency*. Then we discuss some examples of the reasoning tasks based on the proposed representation structure. A harmonisation example produced using the framework discussed is given to illustrate our points.

1 BACKGROUND

The physical properties of our auditory system are well understood. However, we still do not fully understand the psychoacoustic properties of the system. The understanding of how we conceptualize and perceive the world through sound is still vague at present. Despite these ambiguities, we wish to discuss the musical *concept representation* for computation using computers.

The approach we have taken is a symbolic approach which offers an effective problem solving approach, especially in the knowledge intensive style. In this approach, the system

- can work well with a reasonable number of rules (Ebcioglu's CHORAL system has only 300 plus rules). Hence, it is compact. Other approaches, such as the probabilistic approach, would need a huge training data to be able to perform at the same level;
- supports an explicit coding of heuristics and meta programming, etc, which is very important since these tactics when carefully modelled could help when dealing with intractability problem; and
- supports explanation since the knowledge in the sys-

tem is structured (an example of unstructured knowledge is an artificial neural network model).

We discuss the idea of concepts and their representations next.

1.1 Concepts

A concept denotes an idea of a class of objects [8]. For examples; red, yellow, the sun, etc are concepts. Examples of musical concepts are *tempo, cadence, pitch name, pitch frequency*, etc.

The representation of concepts in our work are expressed using logical *terms*. Concepts may be declaratively or procedurally expressed. Procedurally expressed concepts have some advantages in the sense that control structures (which come with the procedures) may be imposed on the manipulation of concepts. Heuristics could be included in the control structure.

1.2 Representations

Many researchers investigate the music representation for computations using computers. Here, we choose to classify them into four categories as follows:

- Notation typesetters: This class of representation facilitates the typesetting of musical symbols. *Musix-TeX* [14] is an example of representation framework in this class.
- Sequencers: This class of representation facilitates the storing and playing back of musical events. Musical events are represented at sound pressure level (i.e. represent the true wave) or at an abstract pitch and time levels. *Wave* and *Midi* specifications are two examples of this representation class.
- Synthesizers: This class of representation focuses on sound generation. It also facilitates the storing and playing back of musical events. Musical events may be represented at the note level or at a finer

grain size (depending on the synthesis routine). The *Music-N* family and *Csound* are representative examples of this class. The *Csound* represents music at a signal processing level. This allows a direct access to a lower level than the note level.

- **Inferencers:** The term inferencers used here refers to a class of computation which places more emphasis on the symbolic inference process. In this class, the representation must support symbolic computation at the desired conceptual level (Interested readers may see [2, 3, 12, 15, 16]).

There is no such thing as “a perfect representation framework”, since different frameworks have different advantages and disadvantages. For examples, the midi specifications adequately represent performing information (e.g. velocity of a key, onset time and duration, etc). However, this midi framework cannot distinguish between $c4\sharp$ and $d4\flat$, as both share the same midi note number (i.e. number 61). This drawback comes from the expressiveness of the representation framework itself. To overcome drawbacks inherent in the chosen framework, the representation framework should be transformable to other frameworks [6]. We use the term *multiple views* to denote this idea.

The abilities to see, hear and edit musical ideas are essential interfaces between human and music applications. Hence, notations typesetters, sequencers and synthesizers are common features and many commercial packages support these concepts. However, symbolic computations are not common in commercial packages. This is mainly due to the varieties and levels of the reasoning tasks required for different applications. For example, the transposition of a scale is a much simpler task than the filling of a cadence. Most commercial packages can perform the transposition but not the more complex tasks of reasoning about cadence properties.

Since complex tasks require expressive representation, the design of a representation framework is a very important issue. This leads us to the discussion of the representation of concepts used in symbolic computation tasks.

2 AIMS

The representation must support the reasoning we normally undertake (or believe we undertake). In other words, we would like to have the representation system that is native to our way of seeing and solving problems. Hence, there should be a minimal distortion in the mapping between the problem and the corresponding computational model, since this would allow us to express our problems and solutions at a level which is natural to our problem solving skills.

We illustrate this by examining the tasks required in analysing a piece of work and highlighting the corresponding computational tasks in computers. This establishes how we should represent musical concepts in computers. We then highlight issues related to representation: *abstraction, expressiveness, multiple-views* and *reasoning efficiency*.

2.1 Computational Musicology

A piece of music can be analyzed from different perspectives. The activities involve narrating musical events; describing observed patterns; logical reasoning and numerical computing. The task of analysing usually aims to reveal syntactical structures and their associated meanings (which could be concept related to musical structures such as cadences, intervals; or concepts which are more subtle such as anger, happiness, sadness, etc).

The excerpt below (only the first 9 bars) is a classic short piece from Schumann’s *Kinderscenen*, op. 15, titled *Träumerei*. We shall analyse the piece and then reflect on the analysis tasks we have performed. The aim of the exercise is to highlight the nature of the analysis tasks.

The image shows a musical score for the first 9 bars of Schumann's 'Träumerei'. It consists of three systems of two staves each (treble and bass clef). The first system includes a tempo marking '♩ = 100' and a dynamic marking 'p'. The score features a variety of note values, rests, and phrasing slurs. The second system ends with a 'ritard' marking, and the third system ends with a 'cut...' marking.

Narrative and descriptive




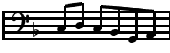
The leading figure of this narrative and descriptive style is, perhaps Donald Tovey (1875-1940). In this style, analysts describe the work; narrate the work; compare the work with and contrast the work against other works. However, analysts do not usually interpret the work with extra-musical contents. The above music example would be described (in this style) as follow:

The piece is from Schumann’s collection of short pieces from the album titled *Kinderscenen*. The piece is song number 7 of the to-

tal 13 songs. The piece is in the key of 'F major'. The first four-bar phrase progresses from tonic to subdominant, dominant and tonic before a closing in dominant. Schumann intentionally delays the accent (i.e. lengthen the notes, changes harmony) to the second beat of every bar. Prominent melody is in the top line all the time until the end of the phrase where the bass joins the first phrase and the second phrase together; and so on.

Analyse Structural Patterns

In this style of analysis, analysts only describe structures and patterns observed in the work. The observed structures/patterns may or may not be associated with other concepts (i.e. semiotics). This is similar to the descriptive style. However, this style tends to be more quantitative and, therefore, is very suitable for computers [1]. For example, the following musical patterns could be counted according to different criteria.

Patterns	Example of criteria
	exact match of interval
	exact match of interval/duration
	match similar figure
	match similar figure

Analyse extra-musical Contents

In this style of analysis, analysts are willing to give more interpretation of the music materials. This is common in the analysis of programmed music. The programmed contents are not totally groundless since many composers actually set their music with narrative scenes. An example of analysis in this style could be found in the analysis of Beethoven's Pastoral symphony and the Ninth symphony (see [4, 9]).

To capture and facilitate reasoning behaviours required in the above tasks, our representation must be expressive enough for the domain. As we have mentioned, there is no such thing as a perfect representation for all purposes. As such, the framework should support *multiple views* so that the representation could be in a structure most suitable for the computation task.

2.2 Abstraction, Expressiveness, Multiple views and Reasoning efficiency

Abstraction and Expressiveness

Abstraction and expressiveness are two important issues which we must consider in designing our representation framework. This is because we cannot include all the concepts we want in our universe of discourse due to intractability; so we have to be selective. We abstract the domain so that it is expressive enough for us to say things we want to say and abstract enough to be computationally feasible.

Assuming we have decided to represent music at the abstraction level of pitch and time, there are many plausible predicates that represent music from these two dimensions (of pitch and time). For example, a pitch may be encoded in terms of *Height, Chroma, Position in the circle of fifths* or *Name, Register* or *Key positions* in a piano while time may be encoded in terms of *physical time, duration, beat* or *rhythm*. The main concern in this concept formation task is that the chosen primitives and representation language must be expressive enough to allow discussions about the domain.

It is our goal to represent music and to carry out musical analysis in somewhat conventional perspectives. We list examples of primitives below:

- Pitch-related attributes (e.g. Name, Accidental, Register, Frequency)
- Sonority-related attributes (e.g. Intervals, Chords)
- Performing instruction-related attributes (e.g. Forte, Piano, Accent)
- Structure-related attributes (e.g. Key, Metrical structure, Phrases, Cadences)
- Process-related attributes (e.g. Order of tasks)
- General information attributes (e.g. Genre, Piece name)
- Physical time (e.g. Clock time)
- Notated musical time (e.g. Pitch duration, tempo, beat)

We construct knowledge of musical structure based on these primitive propositions (e.g. *pitch name, pitch duration, pitch register, time, etc*). For example: the predicate $pitch(1, nat, 4)$ may represent the pitch equivalent to the *middle c* on a piano.

Multiple views

We usually work with different representation shapes in any non-trivial problem solving process. Multiple views are different representation shapes of the same concept.

Multiple views are the mapping between different representation shapes and the base representation shapes ($f : views \rightarrow base_viewpoint$).

Reasoning efficiency

We sometimes find that it is more natural to code a piece of knowledge in procedural form [13]. For example, a concept of musical cadences could be declaratively stated. We may declaratively describe different cadence types as follows:

$\forall x, y \text{ dominant}(x) \wedge \text{tonic}(y) \rightarrow \text{perfect}(x, y)$
 $\forall x, y \text{ dominant}(x) \wedge \text{submediant}(y) \rightarrow \text{deceptive}(x, y)$
 $\forall x, y \text{ tonic}(x) \wedge \text{dominant}(y) \rightarrow \text{imperfect}(x, y)$
 $\forall x, y \text{ subdominant}(x) \wedge \text{tonic}(y) \rightarrow \text{plagal}(x, y)$

However, the same knowledge could be captured in a procedural style (as a function in this example).

```

harmony ← harmonyType(sound)
cadence ← cadenceType(harmony,harmony)
sound    x,y
harmony  a,b

```

```

function progression(x,y)
{
    a = harmonyType(x);
    b = harmonyType(y);
    return cadenceType( a,b )
}

```

It therefore seems appropriate that we should allow procedural construction of primitive musical structures. Both declarative musical structures and procedural musical processes are hierarchical in nature. They may be constructed from many lower level structures and/or processes. Examples of procedural primitives are predicates such as *subtract_pitch*, *add_pitch*, etc; control structures in the language such as *and*, *or*, *if-then*, etc.

We summarise that the representation of the domain knowledge must observe the following concepts:

- **Abstraction and Expressiveness:** The main purpose for this is to limit the size of the universe of discourse while still having enough vocabulary for the problem solving process. This is also dependent on the nature of the problem. For examples, formal analysis would require that the system should be able to talk about the knowledge of musical structures (e.g. pitch, melody, harmony, texture, form, cadence), however, frequency of the note may not be so important in this case and should be abstracted.
- **Multiple views and Reasoning efficiency:** If some information is abstracted from the representation, we could always get the information provided there

are enough clues (either implicit or explicit knowledge in the system). The ability to move among different representation forms is very useful as it could enhance reasoning efficiency since different tasks may require different representation shapes. Reasoning efficiency may be improved from the control structure [10].

3 MAIN CONTRIBUTIONS

3.1 Describing Concepts

Representing musical concepts using logical language facilitates predication of concepts. The predicates chosen must be expressive enough to allow discussions of music theories at the appropriate level of the task (some examples of concepts are highlighted in Figure 1, such as tonic, dominant, anacrusis, melodic figures, etc. We do not label them in the figure for readability).

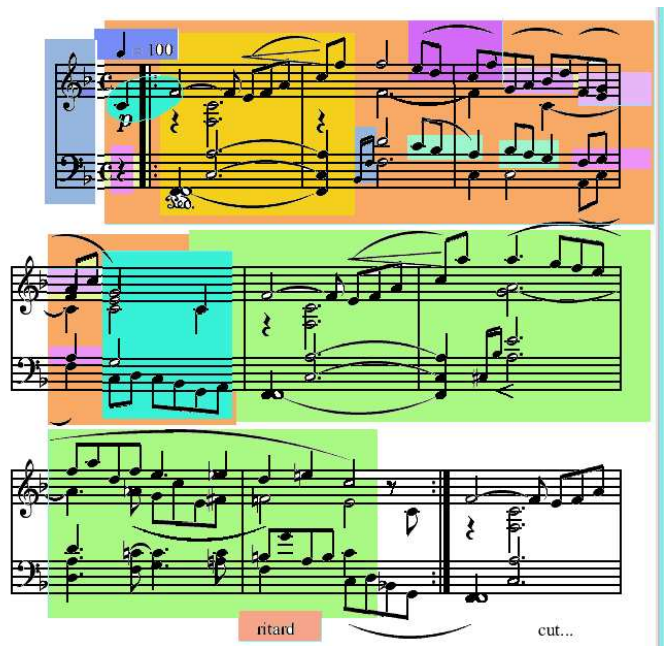


Figure 1: Highlighted patterns are concepts

Descriptions of musical knowledge in terms of: *form*, *melody*, *harmony*, *texture* are common practices. This covers the concepts of *pitches*, *modes*, *keys*, *triad chords*, *seventh chords*, *functional tonalities*, *modulations*, *simple ornamentations*, *melody*, *forms of musical works*, etc. We give some examples of concepts below:

Describing form, melody, harmony and texture

The formal structure of works in a particular genre abides to the common patterns. Examples of the concepts rele-

vant to formal song structures are given below (based on the song *Triumfarei*).

- Time signature is $\frac{4}{4}$, key signature is F major.
- The layout of the song is A–B–A' with a repeat of the first two phrase (i.e. the A section).

The contour of the melody, its rhythmic shape, the harmonic movement, the texture of the sound mass etc, are important features of a work. Examples of concept in this area are:

- all phrases start with anacrusis.
- the harmonic progression of the first phrase is: I–IV–I_c–V₇–I–V.
- all phrases are held together by similarities in melody line and are contrasted by texture and harmonic colours.

During the problem solving process, concepts related to the domain (i.e. form, melody, harmony, texture) are generated, added to or deleted from the system.

3.2 Describing concepts using logical language

A logical language provides a natural framework for representing knowledge asserted in natural language. A brief description of some primitive categories in the language is given in Figure 2. Our knowledge representation is based on a logical approach. The knowledge of the world is represented as an atomic sentence which may be constructed with appropriate logical connectors. In brief, we may write our knowledge in a Horn clause: $A \leftarrow L_1 \wedge L_2 \wedge \dots \wedge L_n$ where L is a literal and A is an atomic sentence. Examples of concepts described in a logical language are given below:

- Pitch: Pitch is an abstract representation of a set of tones. At this level the distinctions between pitches are in their orders which could be referred to as different frequencies or names. For example, we may represent a pitch as a tuple:
 $\langle \text{pitchName}, \text{accidental}, \text{register} \rangle$;
i.e. $\text{pitch} \rightarrow \{1\ 2\ 3\ 4\ 5\ 6\ 7\} \times \{\text{nat}\ \sharp\ \flat\ \text{bb}\ \text{x}\} \times \text{Integer}$ [11].
- Notated musical time: We quantify the representation of time used to quantify the length of pitches (e.g. duration) or other properties (e.g. beat). Time, for instance, may be represented using integers. The smallest unit of 1 could signify a demisemiquaver (thirty-second note).
- Note event: An event constructed from pitch, onset time and duration time could be predicated as $\text{event}(\text{pitch}, \text{onset}, \text{duration})$.

Categories	Description
Constant	
34	An integer constant '34' (which could mean anything apart from the quantity 34)
0.6	A float constant '0.6'
'C major'	An atomic constant 'C major' corresponds to a term used in music theory.
Variable	
Cadence	A variable stands for some definite but unidentified object.
Compound terms & Description	
$\text{scale}(\text{'C major'}, [c, d, e, f, g, a, b])$ The scale of C major has the pitch class of {c d e f g a b c}.	
$\text{perfectCadence}(0.8)$ There are 0.8 probability that the cadence is a perfect cadence.	
$\text{chromaticInterval}(\text{min-third}, 3)$ A database entry indicates a number of semi-tones and their equivalent name (a minor third interval).	
$\text{chord}(\text{dominant}, \text{root}(\text{?Pitch}))$ A variable pitch is a root of a dominant chord.	
$\text{gtpitch}(+\text{Pitch1}, +\text{Pitch2})$ A function that returns 'yes' if Pitch1 is higher than Pitch2 and return 'no' if Pitch2 is higher than Pitch1.	

Figure 2: Example of Terms used in the system

Definition 1 Concept protocol: A concept protocol is a collection of concepts. The construction of a protocol could be via prior knowledge or through a learning process.

Definition 2 Concepts: A concept is a collection of attribute-value pairs. A concept is classified with the same name to the protocol it is most similar to. Attributes and values are terms (t_i).

$$\text{concept} = \{(att, val)_1, (att, val)_2, \dots, (att, val)_n\}$$

The value may either be declaratively or procedurally specified using primitive terms together with control structures (c_i) available in its own language (\mathcal{L}).

$$val \subset \{t_i, c_i\}^*$$

Concepts may be constructed from other concepts as well (i.e. the values could be the collection of concepts under specified constraints).

At implementation level, concepts are usually constructed in certain shapes (i.e. data structures) which are natural to the abstract machine being used.

3.3 The concept of the conventional Score

In this section, we discuss how the real world musical notations are represented in computers. We use the term

Score to denote the representation of knowledge of the real world score. The score concept contains a large amount of other musical concepts. Those concepts are represented in two main components: *musical materials* and *interpretations*. We notate the score using a constructor:

score(Musical materials, Interpretations)

The musical material part represents knowledge in a shape suitable for describing the problem statement. The interpretations part represents knowledge in different viewpoints (depending on the tasks). The base music materials can be shaped and specialised to suit different computation models. New pieces of knowledge generated during the problem solving process may be added to or deleted from the interpretations. This leads us to a detailed discussion of music materials and interpretations.

3.3.1 Music materials

Music materials is a collection of concepts. Here we choose the concept of musical line as a surface representation. The decision to choose the representation shape as a surface representation depends heavily on the problem statement. For example, a line is a perfect surface representation if the problem statement describes a melody line. Let us discuss the examples of some important concepts, which we notate here as attribute-value matrices (AVM) [7]. The AVM used here has one feature which is unique from the notion of AVM used in most linguistic works. Here, the value of an attribute could be a program.

Definition 3 Pitch: *Pitch notations are based on the conventional western music notations. We could represent a pitch as a tuple $\langle \text{pitchName}, \text{accidental}, \text{register} \rangle$.*

$$\begin{bmatrix} \text{type} & \text{pitch} \\ \text{name} & \text{middle_c} \\ \text{shape} & [1, \text{nat}, 4] \\ \vdots & \vdots \\ \text{att}_n & \text{val}_n \end{bmatrix}$$

Definition 4 Note events: *A note event e describes pitch and time information. Other pieces of information included at this level are those related to the notation (e.g. accent, staccato, trill, appoggiatura, fermata, etc.) of each pitch event at the local level.*

$$\begin{bmatrix} \text{type} & \text{note_event} \\ \text{name} & \text{crotchet} \\ \text{shape} & \begin{bmatrix} \text{type} & \text{pitch} \\ \text{name} & \text{middle_c} \\ \text{shape} & [1, \text{nat}, 4] \\ \vdots & \vdots \\ \text{att}_n & \text{val}_n \end{bmatrix} \\ \text{onsetTime} & 0 \\ \text{duration} & 8 \\ \vdots & \vdots \\ \text{att}_n & \text{val}_n \end{bmatrix}$$

Definition 5 Line: *Line is a concept describing a sequence of note events.*

$$\begin{bmatrix} \text{type} & \text{line} \\ \text{name} & \text{soprano} \\ \text{shape} & [e_1, e_2, \dots, e_n] \\ \vdots & \vdots \\ \text{att}_n & \text{val}_n \end{bmatrix}$$

Declaring shape using procedural knowledge

We have mentioned earlier that sometimes it is more natural to code knowledge in procedural form. Examples of the interval concept below should illustrate the point. Here the shape is actually a logical program.

Definition 6 Interval: *Interval is a concept describing the distance between two pitches.*

$$\begin{bmatrix} \text{type} & \text{interval} \\ \text{name} & \text{perfect_fifth} \\ \text{shape} & \text{sub}_{pp}(\text{pitch}_1, \text{pitch}_2) \rightarrow 7_semitones \\ \vdots & \vdots \\ \text{att}_n & \text{val}_n \end{bmatrix}$$

3.3.2 Interpretations

Interpretations is also a collection of concepts. Different concepts could be assigned to the same music materials to create multiple views. Actually, the concepts in the *music materials* and in the *interpretations* are of the same class. The concepts in music materials are the form of concepts which we have decided to bring to the surface as the base representation. The concepts in interpretations are concepts which we usually add in, or delete from during the problem solving process.

Interpretations are grouped under different general concepts such as form, melody, harmony and texture. The groupings are arbitrary but must be consistent when adding new concepts to the interpretations.

$$\text{Interpretations} = \{\text{Concept}_1, \text{Concept}_2, \dots, \text{Concept}_n\}$$

For example, a given melody line may have more than one views associated to it. Ebcioğlu, in *CHORAL*, introduces the following views: *the chord skeleton view, the fill-in view, the melodic-string view, the merged melodic-string view and the time-slice view* [5].

3.4 Reasoning with musical concepts

Symbolic reasoning involves the manipulation of symbols. We need to be able to manipulate knowledge contents in our score representation. The basic operations are the abilities to create new concepts, retrieve, delete and edit existing concepts.

Forming new concepts

New concepts could be created from a set of existing concepts. For example a set of pitches being grouped under specific constraints could be viewed as a chord; or a melody line; or a motive, etc. There are some issues we need to consider when we create new concepts from a collection of existing concepts.

- **Well-formed structure:** We impose prior knowledge by specifying the well-formed structures of interesting musical structures. This abstraction limits unfruitful combinations of concepts. The grouping of two or more concepts depends on the nature of the new concept being created. The grouping must conform to the grouping grammar (e.g. a note event is formed from pitch and time, a line is a collection of note events, etc).
- **Compatibility:** Two groupable concept may share some attributes. The shared attributes must be compatible, e.g. the attribute-value pair $(key, 'F\ major')$ is not compatible with the pair $(key, 'G\ major')$, but is compatible with (key, Key) or $(key, 'F\ major')$.

Comparing concepts

A concept may be noted as a directed graph. Attributes are edges and values are leaf nodes of the graph. The figure below shows an attribute-value matrix and the equivalent directed graph. The common form of concept comparison is *unification*. Measuring *similarity* is another important comparison technique. One way to justify the similarity between concepts is to compare the size of the *isomorphic sub-graph* that they share. However, each attribute-value pair does not carry the same significant weight. So in practice, each concept protocol should have its own comparison procedure.

Transforming concepts

There are some basic building blocks which are well developed in music theory (e.g. pitches, intervals, scales, chords). Hence, it is possible to manipulate them using generic operations. Examples of these operations are:

- **The basic operation of time:** Time dimension is the landmark of note events. Basic arithmetic on the time is an essential requirement. For examples, add_{xy} and sub_{xy} refer to add and subtract operations, where x and y are each of $\{p\ i\ t\ d\}$ standing for pitch, interval, time and duration respectively [16].

```

adddd : Duration × Duration → Duration
addtd : Time × Duration → Time
subtt : Time × Time → Duration
subdd : Duration × Duration → Duration

```

<i>type</i>	<i>note_event</i>						
<i>name</i>	<i>crotchet</i>						
<i>shape</i>	<table border="1"> <tr> <td><i>type</i></td> <td><i>pitch</i></td> </tr> <tr> <td><i>name</i></td> <td><i>middle_c</i></td> </tr> <tr> <td><i>shape</i></td> <td>[1, nat, 4]</td> </tr> </table>	<i>type</i>	<i>pitch</i>	<i>name</i>	<i>middle_c</i>	<i>shape</i>	[1, nat, 4]
<i>type</i>	<i>pitch</i>						
<i>name</i>	<i>middle_c</i>						
<i>shape</i>	[1, nat, 4]						
<i>onsetTime</i>	0						
<i>duration</i>	8						
<i>function</i>	<i>dominant</i>						
<i>keySignature</i>	'Fmajor'						
<i>instrument</i>	<i>piano</i>						
<i>volume</i>	120						

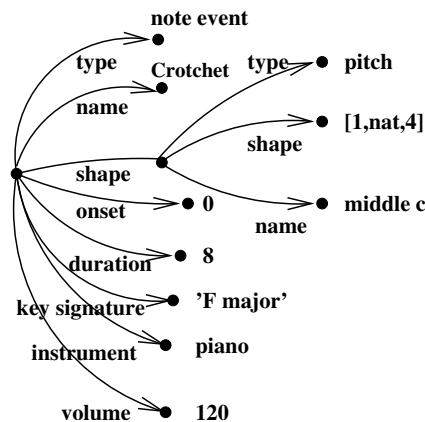


Figure 3: An AVM and a directed graph of a concept

- **The basic operation of pitches:** Pitch comparison is the basic operation of a pitch class. To compare the height between pitches (e.g. C4 is higher than C3). Basic comparisons are greater than; less than; equal to; greater than or equal to; less than or equal to [16].

```

eqpp : Pitch × Pitch → Boolean
gtpp : Pitch × Pitch → Boolean
subpp : Pitch × Pitch → Interval
subpi : Pitch × Interval → Pitch
addpi : Pitch × Interval → Pitch
subpp : Pitch × Pitch → Interval
subpi : Pitch × Interval → Pitch

```

Operations by means of rules, tests and measures

The generic operations tend to be reusable when the knowledge content is viewed from the same perspective. The combination of these operations is limitless. However the behaviour of a single operation or their compositions may be classified into three main categories:

- Operations which yield boolean true or false. These operations are classified as *tests*.
- Operations which yield measurements. These operations are classified as *measures*.

- Operations which yield a concept output. These operations are classified as rules.

Assuming that we are looking for a plausible next pitch from a given a pitch, we could construct an operation which gives a set of plausible progressions from the given pitch (rules). Then the answer may be quantified according to some specifications (measures); or filtered according to some criteria (tests).

4 EXAMPLES

The idea presented in this paper has been implemented in a system that harmonises a given melody input (see [10] for more information). The example below illustrates the harmonisation of Bach's chorales titled *O Haupt voll Blut und Wunden*.

The system takes the original Bach's melody and fills in other voices automatically. Of course, to be able to do this, the system must be equipped with the necessary domain knowledge.

The image displays a musical score for Soprano Alto (SA) and Tenor Bass (TB) voices. It consists of four systems of music. Each system has two staves: the top staff is for SA and the bottom staff is for TB. The music is in G major (one sharp) and 4/4 time. The first system shows the SA voice with a melodic line and the TB voice with a bass line. The second system continues the melody with some rests in the SA voice. The third system shows a more complex harmonic texture with both voices active. The fourth system concludes the passage with a final cadence.

5 CONCLUSION

Representing musical concepts in the proposed framework yields expressive representations and supports symbolic computation using computers. The proposed framework is related to the CHARM's hierarchical structures of events, constituents [16, 11]. The proposed framework ex-

tends the idea presented in CHARM by expressing musical concepts using a language framework (where the value of an attribute could be declaratively or procedurally expressed).

Representing a concept protocol as a list of attribute-value pairs is simple, yet it provides an effective means to abstract the domain with prior knowledge. Constructing, comparing and transforming these concepts could be effectively formulated. Multiple views are natural to the way we see problems and tackle them. The system built in this style can perform a complex harmonisation task with a reasonable harmonisation output.

References

- [1] Christina Anagnostopoulou, Dominik Hörnel, and Karin Höthker. Investigating the Influence of Representations and Algorithms in Music Classification. In Geraint Wiggins, editor, *Proceedings of The AISB'99 Symposium on Musical Creativity*, pages 35–41. AISB, 1999.
- [2] Bernard Bel. Symbolic and sonic representations of sound-object structures. In M. Balaban, K. Ebcioglu, and O. Laske, editors, *Understanding Music with AI: Perspectives on music cognition*, chapter 4, pages 65–109. The AAAI Press/The MIT Press, 1992.
- [3] Francis Courtot. Logical representation and induction for computer assisted composition. In M. Balaban, K. Ebcioglu, and O. Laske, editors, *Understanding Music with AI: Perspectives on music cognition*, chapter 7, pages 157–181. The AAAI Press/The MIT Press, 1992.
- [4] Wyn Jones David. *Beethoven Pastoral Symphony*. Cambridge University Press, 1995.
- [5] K. Ebcioglu. An expert system for harmonizing four-part chorales. In S. M. Schwanauer and D. A. Levitt, editors, *Machine Models of Music*, chapter 17, pages 385–402. The MIT Press, 1993.
- [6] David Huron. Design principles in computer-based music representation. In Alan Marsden and Anthony Pople, editors, *Computer Representations and Model in Music*, pages 5–40. Academic Press, 1992.
- [7] D. Jurafsky and J. Martin. *Speech and Language Processing*. Prentice-Hall, 2000.
- [8] Brown Lesley. *The New Shorter Oxford English Dictionary*. Clarendon Press, Oxford, 1993.
- [9] D. B. Levy. *Beethoven the Ninth Symphony*. Schirmer Books, 1995.
- [10] S. Phon-Amnuaisuk. Control language for harmonisation process. In Christina Anagnostopoulou, Miguel Ferrand, and Alan Smaill, editors, *Music and Artificial Intelligence, Second International Conference, ICAI 2002, Edinburgh, Scotland, UK, September 12-14, 2002, Proceedings*, volume 2445 of *Lecture Notes in Computer Science*. Springer, 2002.
- [11] 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 approach; Proceedings of a Workshop held as part of AI-ED 93, World Conference on AI in Education, Edinburgh*. Springer-Verlag, 1993.
- [12] Travis Pope Stephen. The SmOKe music representation, description language, and interchange format. In *ICMC Proceedings 1992*, pages 106–109. The Computer Music Association, 1992.
- [13] W. Smoliar Stephen. Process structuring and music theory. In S. M. Schwanauer and D. A. Levitt, editors, *Machine Models of Music*, chapter 9, pages 188–212. The MIT Press, 1993.
- [14] D. Taupin, R. Mitchell, and A. Egler. *MusixTex: Using Tex to write polyphonic or instrumental music*. <ftp://hplib.lps.u-psud.fr/pub/musixtex/>, 1997.
- [15] R. West, P. Howell, and I. Cross. Musical structure and knowledge representation. In P. Howell, R. West, and I. Cross, editors, *Representing Musical Structure*, chapter 1, pages 1–30. Academic Press, 1991.
- [16] G. Wiggins, M. Harris, and A. Smaill. Representing music for analysis and composition. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI'89)*, 1989.