Computational Music Representation Based on the Generative Theory of Tonal Music and the Deductive Object-Oriented Database

Author(s): Keiji Hirata and Tatsuya Aoyagi

Source: *Computer Music Journal*, Autumn, 2003, Vol. 27, No. 3 (Autumn, 2003), pp. 73-89

Published by: The MIT Press

Stable URL: https://www.jstor.org/stable/3681803

Keiji Hirata* and Tatsuya Aoyagi†
*NTT Communication Science Laboratories
2–4, Seika-cho Hikaridai, Soraku-gun, Kyoto
619–0237 Japan
hirata@brl.ntt.co.jp
†Tsuda College
Tsuda-machi 2–1-1, Kodaira-shi, Tokyo,
187–8577 Japan
aoyagi@tsuda.ac.jp

Computational Music Representation Based on the Generative Theory of Tonal Music and the Deductive Object-Oriented Database

Much conventional music software places very little importance on music theory. These programs often simply manipulate surface structures of music, not the underlying structures handled in theoretical approaches such as the Generative Theory of Tonal Music (GTTM) of Lerdahl and Jackendoff (1983), the Implication-Realization Model (Narmour 1990), and Preference Rule Systems (Temperley 2001). Because music theory can serve as a basis for analyzing and understanding the vague and subjective meanings of music, it should contribute to the development of an intelligent music application. In particular, we think that music theory should underlie a music knowledge representation language to enhance its descriptive power. As a result, users should be able to correctly predict the run-time behavior of a music application.

Many issues must be considered in designing a music knowledge representation language that employs music theory (Dannenberg 1993; Wiggins et al. 1993; Selfridge-Field 1997). Among them, the following point is crucial: software implementation is not a goal of music theory. Because it has been described in neither a formal nor an algorithmic way, the heuristics, such as the rules for a listener's subjectivity and unconscious preferences, are needed to realize working music applications that include key identification, melody segmentation, melodic similarities, metrical analysis, harmonic analysis, and voice separation. To our knowledge, the attempts at computer implementation of music theory at large include Nord (1992), Temperley and Sleator (1999), and Lerdahl (2001). These works are pioneering and significant to music theory and its computer implementation.

Based on the above considerations, we have developed three music applications (Hirata and Aoyagi 1999, 2000; Hirata and Hiraga 2000) in which we combined time-span analysis of GTTM as the music theory with the deductive object-oriented database (DOOD; Yokota 1992; Kifer, Lausen, and Wu 1995) for knowledge representation. (DOOD was originally the name of an international conference and a research field. Here, it only refers to the name of a knowledge representation method.) We think that DOOD has a theoretical foundation and is thus tractable. We assume that GTTM's time-span analysis yields a useful model for music knowledge representation, because GTTM is one of a few music theories that involves the concept of reduction, which plays a fundamental role in our music representation method. It is true that GTTM itself has been a controversial theory of music, but this article is not about proving the validity of GTTM (see, for example, Botelho 1994; London 1994). Also, to write a working program, more efforts for formalizing GTTM are needed than Preference Rule Systems.

The main contribution of our research is to provide a formal framework of music representation and to achieve musical computation based on music theory. Our framework will be helpful in designing a wide variety of musical tasks and algorithms in a consistent way. However, this article primarily describes music representation, not applications of our music representation method. We emphasize here that major work (i.e., a formal rep-

Computer Music Journal, 27:3, pp. 73–89, Fall 2003

resentational theory of music) must be fully developed before computers can automatically derive a time-span analysis from a musical score.

First, let us define homophony and polyphony as used in this article. The conventional definition of homophony is music in which there is a clear-cut distinction between melody and accompaniment/harmony or in which all the parts move in the same rhythmic pattern (Sadie 1980). Polyphony consists of music in more than one part, music in many parts, and the style in which all or several of the musical parts move to some extent independently (Sadie 1980). In this article we introduce formal definitions of these words. *Homophony* means a melody superimposed by subordinate melodies and interpreted as a single melody temporally, and *polyphony* refers to a hierarchical collection of independent homophonies or polyphonies that may be temporally overlapped.

This article is organized as follows. First, we explain GTTM and DOOD. Next, we present the framework for representing polyphony based on GTTM and DOOD and describe the primitive operations for polyphony represented in our framework. Then, we show some examples and discuss the advantages and disadvantages of our framework together with related work. Finally, we conclude by mentioning future work.

Generative Theory of Tonal Music

Lerdahl and Jackendoff (1983) proposed GTTM as a theory for formalizing the intuition of listeners. GTTM provides concepts and procedures for analyzing and interpreting Western tonal music written in common music notation (i.e., on a score). A score is just a surface structure of music, and GTTM derives from the surface structure the score's hidden hierarchical structures that correspond to a listener's intuition represented as the underlying structures. The music treated by GTTM is limited to homophony. GTTM is modeled on Noam Chomsky's transformational generative grammar, as it proposes a finite set of rules for generating many musical pieces.

Outline of GTTM

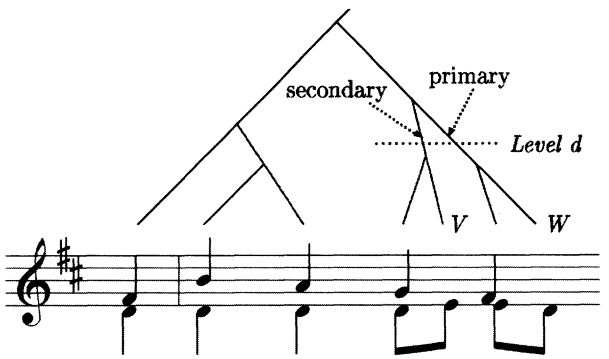
GTTM comprises a *grouping-structure analysis*, a *metrical-structure analysis*, a *time-span reduction*, and a *prolongational reduction*. The grouping structure analysis segments a homophony into nested groups of varying sizes. When we sing a long melody, we must find the proper points for drawing a breath; these points correspond to the boundaries of groups. The metrical structure analysis identifies the positions of strong and weak beats at the levels of a quarter note, half note, whole note, and so on. A conductor moves a baton to conduct the music played, and a listener keeps time by hand clapping or foot tapping; both of these activities correspond in some way to beats.

Time-span reduction represents the intuitive idea that if we remove ornamental notes from a long melody, we obtain a simple melody that sounds similar. An entire piece of Western tonal music can eventually be reduced to an important note or a tonic triad. The time-span reduction is performed based on the results of the grouping structure and metrical structure analyses in a bottom-up manner in the sense that parts come together to form a whole.

Finally, prolongational reduction reflects the musical intuition related to the progression of a piece both at the harmonic and melodic levels. It is usually recognized that parts of a piece or even an entire piece exhibit patterns of tension and release. GTTM provides local structures to support the tension-relaxation pattern: repetition (prolongation) and a change of movement (a "turning point"). The prolongational reduction is performed based on the result of the time-span reduction in a top-down manner. That is, an important note or chord is first selected from an entire homophony in terms of the repetition or turning point, and the piece is then split at that note or chord as a pivot. This step is iterated recursively.

The rules of GTTM comprise *well-formedness rules* (for constructing a tree structure based on analyses) and *preference rules* (the knowledge for the application of the well-formedness rules). The results of time-span reduction and prolongational reduction are obtained in the form of a time-span tree and a prolongational tree. The time-span and

Figure 1. Example of a homophonic passage and a time-span tree, from the first phrase of J. S. Bach's chorale "O Haupt," after Lerdahl and Jackendoff (1983, p. 115).



prolongational trees represent a underlying structure of a piece in the GTTM sense. GTTM basically attempts to look for a unique underlying structure by applying the preference rules as much as possible. However, a piece can be interpreted in various ways, and, indeed, GTTM occasionally derives more than one time-span tree and prolongational tree.

Time-Span Reduction

We formalize time-span reduction in the following manner. A time-span tree is built from the bottom up. First, notes or chords that are adjacent to each other are made into a group. Among the notes or chords contained in the group, the most stable note or chord in the time span covering the duration of the group (called a "head" in GTTM) represents the group. The degree of stability in terms of the time-span reduction is determined by the results of the grouping-structure and metrical-structure analyses. In applying time-span reduction, less important notes or chords are eliminated earlier. Next, neighboring heads similarly make a group. This process continues until a whole piece makes a group.

Figure 1 depicts a sample homophonic passage and its time-span tree, the first phrase of Bach's chorale "O Haupt" in Figure 5.8 (p. 115) of Lerdahl and Jackendoff (1983).

A time-span tree is a binary tree, and in this article we refer to an important branch at a node as *primary*, depicted as a stem, and the other as *secondary*, depicted as a lateral line. At the terminal level of a time-span tree, notes or chords occur on a score. In Figure 1, under the primary branch, note *W* occurs at the last eighth beat of the first measure, and under the secondary branch, note *V* occurs at the upbeat of the third beat. Because *V* and *W* are adjacent to each other at *Level d*, they make a group. Here we say that *V* is a left-branching elaboration of *W*, and this branching indicates that *W* is more stable than *V*. The head of this group is *W*. In this case, *W* is made the head of the group in the *ordinary* way, which is one of four ways to make a head in GTTM. (The others are *fusion*, *transformation* and *cadential retain*.)

Deductive Object-Oriented Database

As described before, music can be vague and subjective on many levels. Here, vagueness means that the information required for resolving the ambiguity of the GTTM analyses is lacking or only partially given. Hence, the knowledge representation method, especially for music, must be able to treat the situations in which some attributes are lacking or only the type of a value, not a value itself, is given or declared.

Next, subjectivity determines which underlying structure derived from the analyses should be adopted. The underlying structures reflect the user's preferences, and some of them may be contradictory. We think that the case-based reasoning technique (Kolodner 1993) can properly treat such subjectivity. Because cases can serve as both data and rules, we can achieve high readability and efficient inference if we employ a scheme that can represent data and rules in the same form. A typical method amenable to this purpose is first-order logic.

The Deductive Object-Oriented Database (DOOD) is an extension of first-order logic in that it can represent the lacking of attributes and type declaration (Yokota 1992) and has almost the same functions as the feature structure from the knowledge-description point of view (Carpenter 1992). Plaza (1995) claimed that the feature structure is suitable for describing cases. A prominent feature of DOOD is a deductive rule for DOOD terms to formally define any relation, which brings about descriptive efficiency, accuracy, and expand-

ability. Incidentally, DOOD is similar to XML (www.w3c.org/XML) and RDF (www.w3c.org/RDF) owing to its object-attribute structure (Castan, Good, and Roland 1999).

Object Term

The DOOD framework is motivated by the insight that real-world entities can be represented as the combination of a basic (atomic) object and a set of attributes. Hereafter, we identify an object term with an object itself. We write an object term as $o(\ldots, l : v, \ldots)$, where o is an atomic symbol for a basic object, $l : v$ is an attribute, l is an attribute's name (label), and v is an attribute's value. We introduce for convenience another notation: for an object term $p(\ldots, l : v, \ldots)$, then $p.l$ is equal to the attribute value v.

Subsumption Relation

The most fundamental relation among real-world objects is the "is_a" relation, and it is modeled as the subsumption relation defined by the deductive rule in the DOOD framework. That is, the subsumption relation (written as $\subseteq$) represents the relation "a more informative object $\subseteq$ a less informative object." In other words, it represents "an instantiated object $\subseteq$ an abstract object" or "a specific object $\subseteq$ a general object."

The subsumption relation is defined in the form of the deductive rule as follows. For two objects $o_1 = p(\ldots, l_m : v_m \ldots)$ and $o_2 = p(\ldots, l_n : \mathrm{v}_n \ldots)$, the subsumption relation between o_1 and o_2 is defined as

$$o_1 \subseteq o_2 \leftarrow \forall\, l_n\, \exists\, l_m\, (l_n = l_m \wedge o_1.l_m \subseteq o_2.l_n)$$

This definition of the subsumption relation means that if for all attributes of o_2 the values of attributes of o_1 are more instantiated, then o_1 is considered more instantiated than o_2 (or o_2 is more abstract than o_1). For example, an object that has fewer attributes is considered more abstract. Between two objects that have distinct basic object terms, the subsumption relation does not hold. By this definition, the subsumption relation can be proven to be transitive.

We show a musical example of the subsumption relation. Consider three objects, *note*, *note(pitch : C)*, and *note(pitch : C, octave : 4)*, where *note* is the concept to represent a single abstract note, and it is extensionally equivalent to the set of all notes. The *note(pitch : C)* is the note only with the information of pitch class, and it is more instantiated than *note*. When we say "a C-major chord consists of notes the C, E, and G," note C is represented as *note(pitch : C)*. The *note(pitch : C, octave : 4)* is more instantiated than *note(pitch : C)* and represents the C4 key on the piano keyboard, for example. Due to the deductive rule of the subsumption relation, *note(pitch : C, octave : 4)* $\subseteq$ *note(pitch : C)* and *note(pitch : C)* $\subseteq$ *note* both hold. Because transitivity also holds with respect to the subsumption relation, we can also write *note(pitch : C, octave : 4)* $\subseteq$ *note*.

Next, we define the subsumption relation between the sets of objects. Assume O_i is the set of objects o_{ij} ($i = 1, 2$ and $j = 1. \ldots N_i$). We can think of various definitions for the subsumption relation between O_1 and O_2. Typical ones in the deductive rule form are

$$O_1 \subseteq_{\mathrm{H}} O_2 \leftarrow \forall s \in O_1\, \exists\, t \in O_2\, s \subseteq t$$
$$O_1 \subseteq_{\mathrm{S}} O_2 \leftarrow \forall s \in O_2\, \exists\, t \in O_1\, t \subseteq s$$

The relation $\subseteq_{\mathrm{H}}$ represents the so-called Hoare order of sets, and $\subseteq_{\mathrm{S}}$ the so-called Smyth order. For example, assuming $b_1 \subseteq b_2$ and $d_1 \subseteq d_2$, we then have $\{b_1,d_1\} \subseteq_{\mathrm{H}} \{a,b_2,c,d_2\}$ and $\{a,b_1,c,d_1\} \subseteq_{\mathrm{S}} \{b_2,d_2\}$. If there is a disjunctive relation between the elements of a set, the Hoare order is appropriate for comparing the sets, and if the conjunctive relation, the Smyth order is appropriate.

Representation of Polyphony

When we design a music knowledge representation method using DOOD, a fundamental question is raised: To what concept in music theory should the subsumption relation of DOOD correspond? In other words, what does the "is_a" relation mean in music? We assert that the counterpart of the subsumption relation is the time-span reduction in GTTM, because the subsumption relation in general represents the instantiation-abstraction relation, and the time-span reduction associates an

instantiated melody with an abstract one. We believe that several examples shown later support this assertion appropriately, and hence this correspondence is the most natural.

That GTTM treats only homophony presents a major limitation, as most Western tonal music is polyphonic. Therefore, we now extend the time-span reduction of GTTM to treat polyphony.

Time-Span Reduction of Polyphony

According to the definition of polyphony, we assume that the time-span reduction of polyphony proceeds as follows. A polyphonic passage is first decomposed to components until they become homophonic (independent streams). The conventional time-span reduction is then applied to each homophonic stream, and the results are finally combined into a single overall grouping structure in some way. Lerdahl (2001) proposed a method for the combination in which proximate homophonic passages are aligned in phase with a metrical grid. Within the aligned homophony, the conventional time-span reduction is applied. We call the method *homophonization by alignment*.

The scope of GTTM is limited to homophony owing to the grouping well-formedness rules and the strong reduction hypothesis (SRH). The SRH claims that a piece is heard in a strict structural manner and that a less structurally important note is heard in association with a surrounding note that is more structurally important. Further, every note or group always belongs to a group one level higher, and, at the same level, the time-spans of any group do not temporally overlap each other. To treat polyphony, this latter rule should be precluded; i.e., time-spans of groups are allowed to overlap. In light of SRH, the homophonization by alignment transforms a polyphonic passage without changing its musical meaning and yields the situation where the latter rule holds.

However, Lerdahl's method raises a problem from the knowledge-representation point of view. That is, we must introduce a new data structure for representing alignment amenable to the DOOD framework. This new data structure may bring about an additional complexity. To ensure our framework deals with alignment as simply as possible, we consider the proximate homophonic passages (in fact, their heads) to be just adjacent branches of a time-span tree. In response, we always place the ordering among these homophonic passages in terms of stability. Figure 2 outlines the time-span reduction of polyphony in our framework.

According to the definition of polyphony, the polyphonic passage in Figure 2 is considered to be a collection of three homophonies temporally overlapped; each homophonic passage is enclosed in a box. Take the left box in Figure 2 that represents a homophony G4–A4–B♭4–E♭5. By time-span reducing the homophonic passage to a single note, an E♭5 is obtained as the head. Similarly, the head of the center passage is C2, and that of the right-most is F2. Then, we determine the ordering among these heads based on stability, which yields the time-span tree one-level higher on the lower side of Figure 2. We will discuss how to set up a head of a time span later in this section.

Representation of the Time-Span Tree

First, we consider two simple melodies, both of which consist of C4 and E4 (see Figures 3a and 3b).

To represent a node of the time-span tree, we introduce a *ts* object, which has attributes *head*, *primary*, and *secondary*. The *ts* object that has only the *head* attribute represents a terminal node of a time-span tree. It stands for a special group consisting of a single note. The following object K_α represents the time-span tree in Figure 3a:

$$K_\alpha = ts(head : C4, \; primary : ts(head : C4), \; secondary : ts(head : E4))$$

Then, we make a time-span tree from K_α by deleting the less-important branch *E4*; the new time-span tree obtained is designated by the following object K_β.

$$K_\beta = ts(head : C4)$$

Here, $K_\alpha \subseteq K_\beta$ holds.

However, the object representing the time-span tree in Figure 3b is equal to that for Figure 3a (i.e.,

Figure 2. Polyphonic grouping. The first two bars of "Autumn Leaves" performed by a jazz pianist. The polyphony is decomposed into homophonic segments, and analysis results are then combined into a single grouping structure.

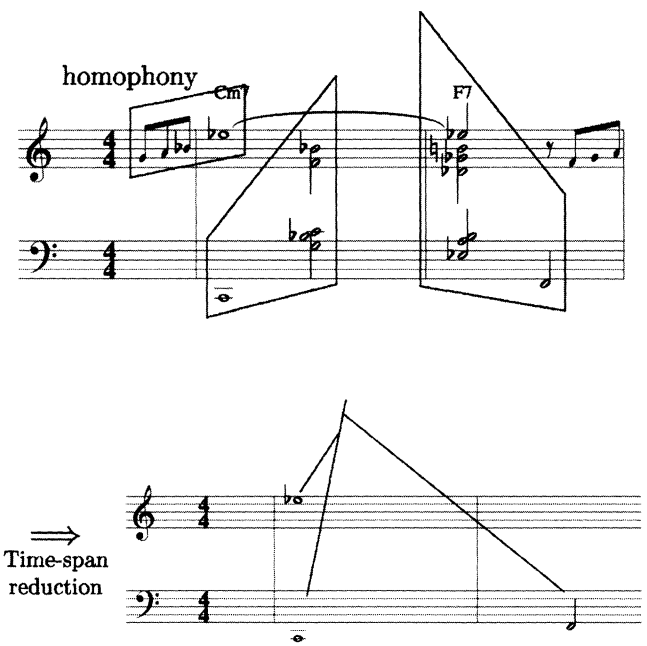
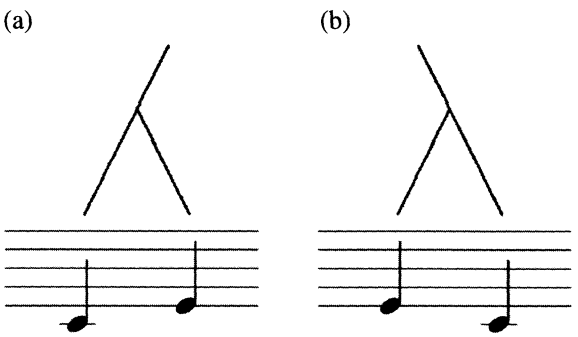


Figure 3. Simple time-span trees that consist of the same pitches.



K_α), because this representation method cannot describe the temporal relation between the primary and secondary attributes. Thus, we must introduce an attribute representing the temporal relation.

Conditions Required for Temporal Relation

To represent the temporal relation for the onset time of a note contained in a polyphonic passage, we impose three conditions. First, the onset time of a relevant note should be given by the difference between the note itself and the more-important nearest note on a time-span tree (Condition 1). Second, abstracting the difference between onset times should provide the ordering information between them, not absolute timings (Condition 2). Finally, abstracting the ordering information should provide the reduced ordering information (Condition 3).

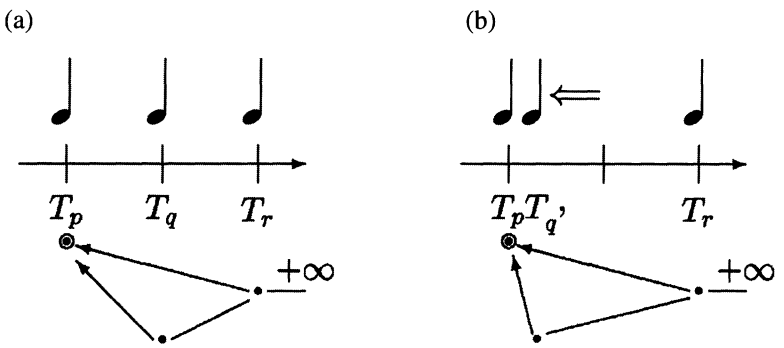
Condition 1 is implicitly prescribed in the definition of the time-span reduction. Condition 2 is motivated by our intuition in terms of homophony. For example, assuming that there are two homophonic passages of the same pitch sequence with distinct durations for each note, we may naturally perceive them as similar to each other, because we understand homophony at an abstract level of pitch sequence. Condition 3 is motivated by our intuition of the abstraction of the ordering. Abstracting the ordering means that the ordering "note N is preceded by note M" becomes the loose ordering "note N is preceded by or occurs at the same time as note M."

Let us consider an example where we make the abstraction of a broken chord (e.g., Alberti bass) and then obtain the chord consisting of the notes of the same pitches. The first half of this abstraction makes the abstraction of the time difference between notes of a broken chord, supported by Condition 2. The second half reduces the ordering to a single chord, which is supported by Condition 3.

Representation of Temporal Structure

To satisfy these three conditions, we represent the onset time of a note by using four attributes: a preceding note, a succeeding note, an adjacent stable note, and a time difference. (We assume that notes surround a relevant note.) Then, one of the temporally preceding notes is considered the preceding note π, and, similarly, one of the temporally succeeding notes the succeeding note σ. Then, the onset time of a relevant note is represented by the constraint that it occurs somewhere after π and before σ, which guarantees the ordering information. Furthermore, we impose the constraint that either π or σ is the head of a group to which the relevant

Figure 4. (a) Temporal structure. (b) Reduced structure.



note anchors. The stable-note attribute indicates whether π or σ is the head. The time difference is the interval between the relevant note and the stable one. If the time difference is given, then the absolute time of the relevant note is obtained, and if not, then only its ordering information is.

Here we assume that if a note or a chord of a group becomes a head by time-span reduction, the onset time of the head is equal to that of the note or the chord. We make this assumption because we want to preserve the amount of information as much as possible in the time-span reduction, although we just deviate from GTTM (discussed again later).

Figure 4 shows an example of a temporal structure based on our framework.

In Figures 4a and 4b, p, q, and r are quarter notes. Assume that the onset time of note p is the origin point of a whole polyphonic passage represented by the symbol $\odot$ in the figures. Here, for the purpose of uniform description, we introduce positive infinite time (denoted as $+\infty$) and negative infinite time (denoted as $-\infty$). Then, the note preceding r is p, and the time of its succeeding note is $+\infty$, because no note succeeds it. Note r occurs on the second beat from p. Similarly, the note preceding q is p, and its succeeding note is r. Note q occurs on the first beat from p. The arrow ($\leftarrow$) in the figure represents the anchoring relation of a relevant note to a head; the source of the arrow corresponds to the relevant note, and the target its *superordinate* note on a time-span tree.

To represent the onset time of a note as a DOOD term, we introduce a new object *temp* that has the attributes *pred*, *succ*, *stable*, and *difference*, which correspond to a preceding note, a succeeding note, a stable note, and the time difference between the relevant note and the stable note, respectively. Here we can think of a couple of methods for designing an object of the temporal structure. In one, to interpret a temporal structure, the structure is traversed from the origin point of an entire piece (note p in Figures 4a and 4b) in a top-down manner. In the other, it is traversed from every note contained in a piece in a bottom-up manner. Our framework employs the latter, because it is desirable that a single object be able to represent a polyphonic passage, and thus the temporal structure for the onset time of every note must be embedded into the structure of a time-span tree as shown in Figures 3a and 3b.

For example, the onset time of note q in Figure 4a, T_q, is written as

$$temp(pred : T_p,\ succ : T_r,\ stable : T_p,\ difference : 1/4)$$

Note that this representation satisfies Condition 1. T_p and T_r are objects for representing the onset times of the preceding note p and the succeeding note r, respectively. Thus, we know that note q occurs after p and before r. The value of the difference attribute *1/4* is the duration of a quarter note. Because $T_q.stable = T_p$, note q occurs one beat after p.

Abstracting T_q with respect to the difference attribute, we get T_s, representing only the ordering information:

$$temp(pred : T_p,\ succ : T_r,\ stable : T_p)$$

Then, we obtain $T_q \subseteq T_s$, which satisfies Condition 2.

Lastly, consider notes q and q' in Figures 4a and 4b as an example for examining Condition 3. Comparing the temporal structure containing q and one containing q', we can consider the latter more abstract, because q' displaces towards p and is eventually reduced to p. This reduction occurs because for note q, p is more superordinate than r on the time-span tree. Hence, we need a new deductive rule for the subsumption relation for this reduction of the temporal structure, which is

$$T \subseteq T.stable \leftarrow T \subseteq temp$$

In the example shown in Figure 4b, because $T_q.stable = T_p$, from the deductive rule we obtain $T_q \subseteq T_p$. When q' is reduced to p, and q' is equivalent to p, it follows that $T_q \subseteq T_{q'}$. That is, the temporal structure in which q occurs is more instantiated than one in which q' occurs (at the position of q), which satisfies Condition 3.

The Syntax of Object Terms

The syntax of objects for representing a time-span tree and a temporal structure is

ts(head : {note},
at : temp| 0,
primary : ts,
secondary : ts)

note(pitch-class : PC, octave : Integer, duration : Rational+)

temp(pred : temp | 0 | −∞, succ : temp | 0 | +∞,
stable : temp | 0 | ±∞, difference : Rational)

Here, *objclass*(*name*$_1$: *objclass*$_1$, . . .) stands for the name of an object class, *name*$_1$ represents the name of an attribute, and *objclass*$_1$ the name or the data type of an object class at the *name*$_1$ attribute. *PC* means a pitch class, such as C, C♯, etc. *Integer* means an integer number, *Rational* a rational number, *Rational+* a positive rational number, and *0* an origin time. The notation "*x* | *y*" means "*x* or *y*", and {*x*} represents a set of *x*.

To hold the temporal structure of an individual note, we introduce an *at* attribute into a *ts* object so that the value of the *at* attribute is a *temp* object. The *head* attribute keeps the set of *note* objects that make a chord. The note object stands for the pitch class, octave, and duration of an individual note of a chord. If the value of the head attribute is a singleton set (i.e., the number of notes is one), it is equivalent to a single note. Since the relation between chord notes is conjunctive, the Smyth order is appropriate for the ordering of the head attribute values. A *duration* attribute is the duration of a chord, and *1/N* is the duration of one *N*th of a measure. For the attributes of a temp object, refer to the previous section.

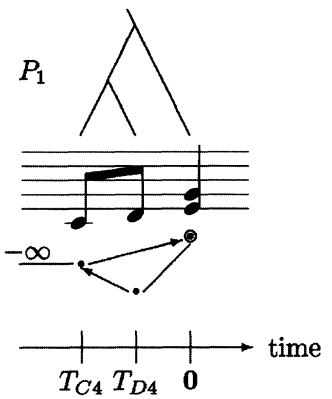
Setting a Head

As mentioned before, to handle polyphony, a head can be placed anywhere within a time span in our framework. There are two ways to set the head attributes (position and duration). One is to place the head at the beginning of a time span and set its duration to the width of the time span, according to the time-span reduction prescribed in GTTM (except for the cases such as *auftakt* and upbeat). We call this a "GTTM head." In contrast, we can set a stable note or chord itself to a head without changing its position and duration. We call this a "conservative head," which has a practical advantage in that less information is lost in the time-span reduction. That is, when a stable note or chord itself becomes the head of a time span, the head preserves part of the information on the actual shape of the polyphony before the time-span reduction. The conservative head plays an essential role when creating complicated polyphony by instantiation (the inverse of reduction): the information of a stable note or the chord to be placed within its time span is retained. Our framework adopts the conservative head rather than the GTTM head.

Example of Simple Polyphony

Figure 5 shows a time-span tree and a temporal structure represented in our framework with a conservative head.

Figure 5. A time-span tree and temporal structure.



The *ts* object for the time-span tree and the temporal structure in Figure 5 is described in the following DOOD terms:

ts[*head* : {N_3, N_4},
  at : 0,
  primary : ts[*head* : {N_3, N_4},
    at : 0],
  secondary : ts[*head* : {N_1},
    at : T_{C4},
    primary : ts[*head* : {N_1},
      at : T_{C4}]
    secondary : ts[*head* : {N_2},
      at : T_{D4}]]]

N_1 = *note*[*pitch-class* : *C*, *octave* : 4, *duration* : 1/8]
N_2 = *note*[*pitch-class* : *D*, *octave* : 4, *duration* : 1/8]
N_3 = *note*[*pitch-class* : *E*, *octave* : 4, *duration* : 1/4]
N_4 = *note*[*pitch-class* : *G*, *octave* : 4, *duration* : 1/4]

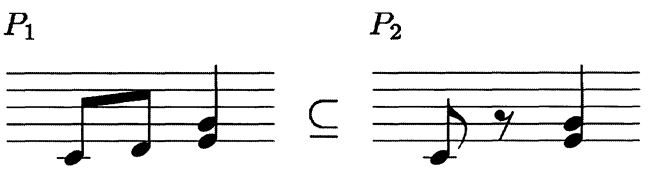
T_{C4} = *temp*[*pred* : $-\infty$, *succ* : 0,
  stable : 0, *difference* : 1/4]
T_{D4} = *temp*[*pred* : T_{C4}, *succ* : 0,
  stable : T_{C4}, *difference* : 1/8]

Here, the *ts* object is shown with intermediate variables N_1, . . ., T_{C4}, . . ., for readability, but the actual *ts* object on a computer may be constructed without these intermediate variables.

Primitives for Polyphony

So far, we have given a formal representation of polyphony. In this section, we introduce three primitive operators for polyphony: subsumption relation ($\subseteq$), least upper bound (*lub*), and greatest lower bound (*glb*). These primitive operators correspond to calculation of the abstraction-instantiation relation, the largest common part, and the superimposition of two polyphonic passages, respectively (Hirata and Matsuda 2002).

Figure 6. A subsumption relation.



First, we briefly discuss the mathematical background for our framework. The subsumption relation is a *partial order*. A partial order is a relation R such that it is reflexive (i.e., $x\ R\ x$), transitive (i.e., $x\ R\ y \wedge y\ R\ z \rightarrow x\ R\ z$), and antisymmetric (i.e., $x\ R\ y \wedge y\ R\ x \rightarrow x = y$). The ordering is partial, rather than total, when there may exist elements x and y for which neither $x\ R\ y$ nor $y\ R\ x$). Then, the domain of *ts* objects makes a complete lattice with respect to the subsumption relation. A lattice is a set in which a partial ordering is defined and all finite subsets have an intersection (*lub*) and a union (*glb*). A complete lattice is a lattice whose every infinite subset has an intersection and a union.

Subsumption Relation

Given a polyphonic passage, the subsumption relation associates it with a reduced or elaborated one with respect to the time-span reduction. Figure 6 shows that, by eliminating the less important note D4, a more abstract polyphony is obtained.

Because we adopt the conservative head, the eliminated D4 is replaced with an eighth rest, and the other notes are unchanged. Hereafter, we sometimes omit a time-span tree and temporal structures for space efficiency.

Figure 7 shows an example of Alberti bass. As the time-span reduction is carried out from P_1 to P_2 to P_3, less important notes are eliminated, and more abstract melodies are obtained.

Figure 7. A subsumption relation of Alberti bass (Mozart, Sonata K. 332, I, m. 1).

We then have $P_1 \subseteq P_2$, $P_2 \subseteq P_3$, and $P_1 \subseteq P_3$, and these relations agree with musical intuition.

Figure 8 shows an example of a melody (Variation III of Mozart's Variations on "Ah! vous dirais-je, maman").

The subsumption relation holds between Variation III and its reduced melody. We performed time-span analysis of this melody using our method, although the time-span trees and the temporal structures are not shown here. We used various ways of making heads; chord C4–E4–G4 at the first beat of the first bar is made in the "fusion" way, for example, as is the chord at the second beat of the first bar. On the other hand, chord G5–C6 at the first beat of the second bar is made in the "transformational" way.

Note that the subsumption relation can be used both actively (generatively) and passively (consumptively). For example, given a polyphonic passage p, we can obtain a new polyphonic passage x such that $p \subseteq x$, and given polyphonic passages p and q, we can check if $p \subseteq q$ holds.

Figure 8. Reduced melody from Variation III of Mozart's K. 265, Variations on "Ah! vous dirais-je, maman" (equivalent to "Twinkle, Twinkle Little Star").

Least Upper Bound

We usually define a primitive $lub(x, y)$ as $min(\{z \mid x \subseteq z \wedge y \subseteq z\})$, procedurally defined as

$$lub(a,b) = \boldsymbol{T} \quad \text{if } a \neq b$$
$$lub(a,b) = a \quad \text{if } a = b$$
$$lub(o, o(l : v)) = o$$
$$lub(o(l : v_1), o(l : v_2)) = o(l : lub(v_1, v_2))$$

Here, a and b are basic (atomic) objects, $\boldsymbol{T}$ represents the top element in the complete lattice constructed from objects, and we assume $o(l : \boldsymbol{T}) = o$. The *lub* operator extracts the largest common part or the most common information of the time-span trees of two polyphonic passages in a top-down manner. If all the heads are made in the "ordinary" way, calculating *lub* involves deleting less stable notes from the two passages until their "pruned" versions are identical.

Figure 9 shows the *lub* calculation of polyphonic passages P_1 and P_3.

In this *lub* calculation, we do not show incomplete objects that do not hold all proper attributes. Note that the *temp* objects of T_{C4} in P_1 and T_{C4} in P_3 are identical.

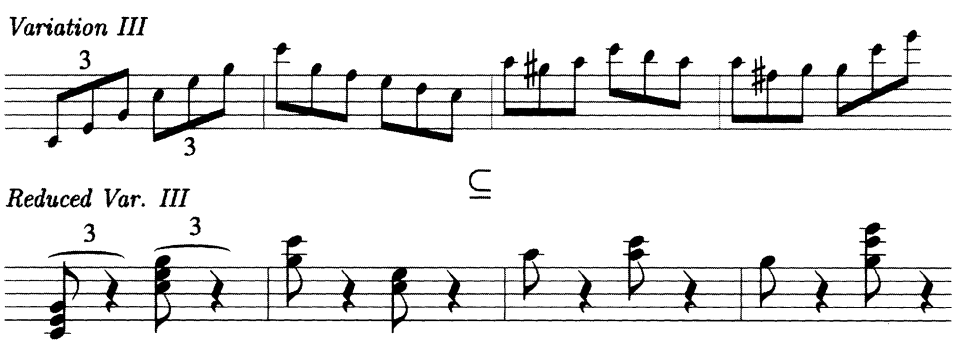
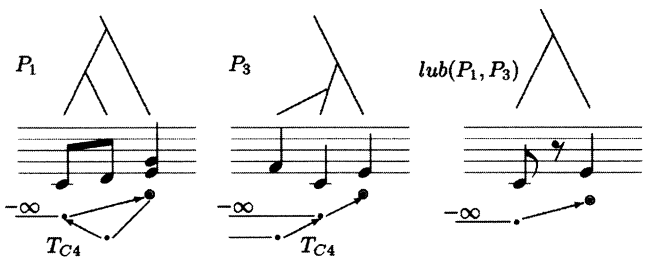


Figure 9. Calculating lub.



In particular, for the *duration* attribute of *chord* objects d_1 and d_2, we introduce the definition $lub(d_1, d_2) = d_1$ if $d_1 < d_2$. The rationale for this is to maintain consistency with the conservative head, in which the onset and duration of a note to be a head are unchanged after the time-span reduction. Here, we can think of another simpler definition: $lub(d_1, d_2) = d_1$ if $d_1 = d_2$, or ***T*** otherwise. Although this definition can also maintain consistency, we do not adopt it, because we think that it loses too much information in the case of ***T***. For example, the duration of note C4 in $lub(P_1, P_3)$ is *1/8*, because $lub(1/4, 1/8) = 1/8$.

Figure 10 shows the *lub* calculation of Variation III and Variation VI from our previous Mozart example.

The *lub* of these melodies yields their most common information. From the subsumption in Figure 8, let us consider the *lub* calculation of the reduced Variation III and Variation VI, where we assume that these have the same time-span trees and temporal structures. For instance, we compare the first chords of these melodies, C4–E4–G4 of duration equal to one-third of a quarter note (*1/12*), and E4–G4–C5 of eighth-note duration (*1/8*). Notes E4 and G4 are common. C4 and C5 are distinct, but pitch class C is common. Therefore, the result of *lub* of the first chords is the chord E4–G4–C with a duration equal to one-third of a quarter note.

Note that "C" represents an abstract note whose pitch class is specified, but not its octave. We say the note is "incomplete." The remaining chords are similarly calculated. According to the above *lub* definition regarding duration, we obtain $lub(1/12, 1/8) = 1/12$.

Figure 10. Calculating lub *and the abstract melody obtained (Mozart, Variations K. 265).*

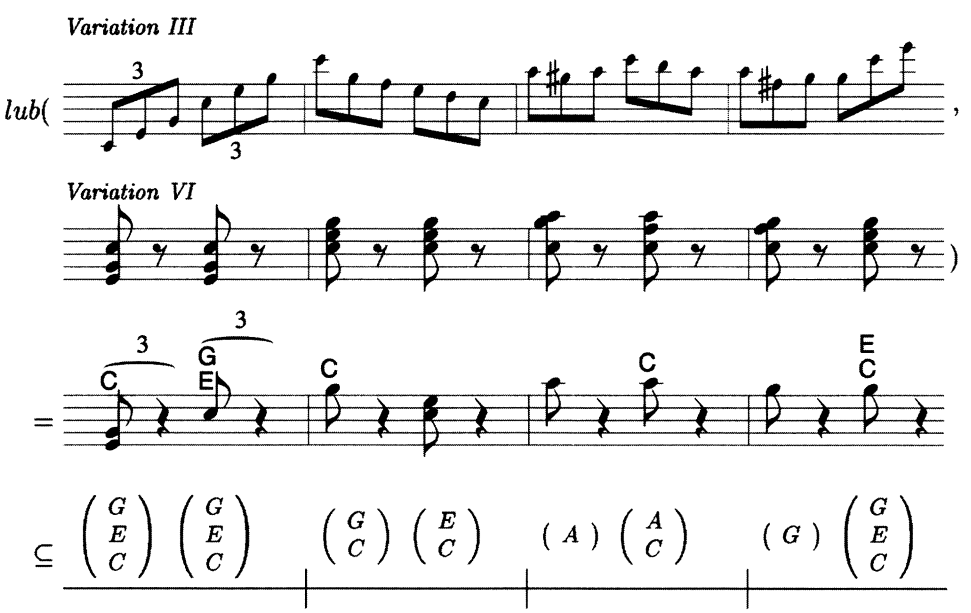
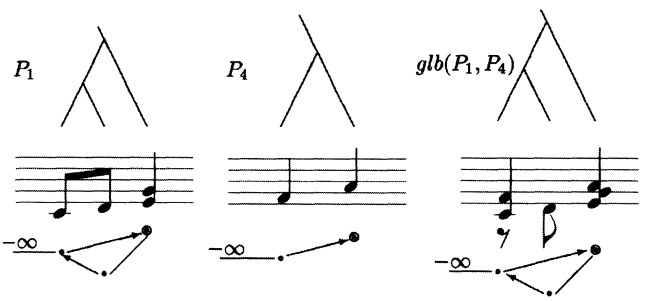


Figure 11. Calculating glb.

Furthermore, the onset-timings of the third melody in Figure 10 are abstracted to the ordering information of the last melody. At that time, the octave information for each note is abstracted, and it is all eliminated. Note that the subsumption relation can associate not only a real melody with another real one, but also an abstract melody of incomplete notes with an abstract melody or a real melody.



Greatest Lower Bound

Similarly, we define $glb(x, y) = max(\{z \mid z \subseteq x \wedge z \subseteq y\})$, and $o(l : \perp) = \perp$. Primitive *glb* joins two polyphonies in the top-down manner as long as the structures of two polyphonies are consistent. If we arrive at a leaf node of either of the two polyphonies, the remaining structure of the other polyphony is expanded at the leaf node. If the value of *glb* of *temp* objects is, for example, $\perp$, then the entire object is also $\perp$ owing to our definition.

Figure 11 shows the sample calculation of *glb* of polyphonic passages P_1 and P_4.

Looking at the time-span trees of P_1 and P_4, we know that the most important events are the last chord in P_1 (E4–G4) and the last note in P_4 (A4). They are joined to E4–G4–A4 in $glb(P_1, P_4)$, and the chord is placed on the origin time ($\odot$). The first note in P_1 (C4) and the first note in P_4 (F4) are left-branching elaborations of their respectively most important events, and they occur one beat before the origin time. Because the positions of these notes are consistent in terms of time-span structure and temporal structure, they can be joined to a chord, C4–F4, in $glb(P_1, P_4)$.

In this *glb* calculation, for the duration attribute, $glb(n_1, n_2) = n_1 \cup n_2$, because we employ the Smyth order between the values of the duration attribute. In particular, for the duration attribute of the chord object, d_1 and d_2, $glb(d_1, d_2) = d_2$ if $d_1 < d_2$, for the same reason as with the *lub* calculation. Note that $glb(P_1, P_3) = \perp$. For the first notes of P_1 and P_4 in Figure 11 (C4 eighth note and F4 quarter note), we have $glb(1/4, 1/8) = 1/4$.

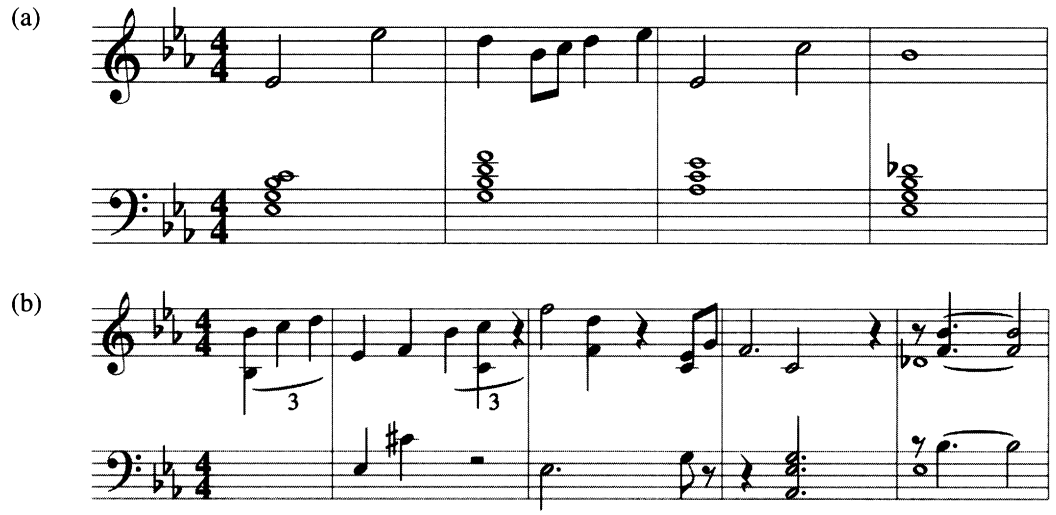
Music theory in general is a framework for understanding music on a score, not for analyzing performance and composition. That is, music theory has provided procedures for analysis (reduction), not synthesis (instantiation). We think the *glb* operator proposed realizes instantiation appropriately.

Some Further Examples

Here we present two examples to which our framework has been applied. The first is an arrangement program. Figure 12 shows the input and output polyphonic passages of our arrangement program that employs case-based reasoning.

The upper half of Figure 12 shows the four-bar input. The input is first divided into four one-bar fragments. For each input fragment, the arrangement program selects the most similar case from a case library, carries out case-based reasoning, and concatenates each output. That is, arrangement proceeds bar by bar. The input consists of a monophonic theme for a right hand and a simple chord progression for a left hand. The output is a solo piano performance for both hands. The sample arrangement in the figure uses as cases the real 32-bar performance of "Autumn Leaves" and a 16-bar performance of "Nefertiti" played by a jazz pianist. The jazz pianist looked at a simple score of "Autumn Leaves" and "Nefertiti" like the upper half of Figure 12 and created improvisations like the lower half, although the entire scores and improvisations are not shown to save space. (The left-hand side of Figure 12 is part of an improvisation.) Thus, the case library here consists of 48 total one-bar cases. In the lower half of Figure 12, we see a couple of *auftakt* (upbeat) notes consisting of B♭4–C5 (D4);

Figure 12. (a) Input polyphonic passage (first four bars of "Over the Rainbow"). (b) Sample output of arrangement application.

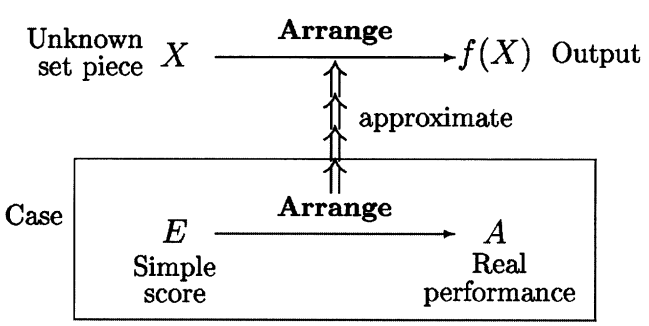


one lies at the very beginning of the score, and the other begins at the third beat of the first bar. These *auftakt*s are originally contained in the case performances used (Hirata and Aoyagi 2001).

We will now briefly discuss the arrangement algorithm. Figure 13 illustrates the arrangement task as an imitation of a function (Hirata and Aoyagi 2000).

In our formalization, the input data comprises an unknown set piece X, a sample arrangement consisting of a set piece E, and the resulting piece of its arrangement A. Here, X, E, and A are all one-bar polyphonic passages. Then, to arrange X, we simply compute $f(X)$ with respect to the function f satisfying $f(E) = A$. (Here we use a single case.) The arrangement $f(X)$ is also a one-bar polyphonic passage. Because not enough information is given to infer the function f, many such functions can be derived. Therefore, we impose some musical constraints onto f and, as a result, we obtain *glb(lub(A,X),E)*, approximating *f(X)* to some extent. The sample arrangement is calculated by *glb(lub(A,X),E)* for each bar. (The standard MIDI file outputs of the arrangement program are available online at www.brl.ntt.co.jp/people/hirata/nttmusicmachines.html#babibun.)

Figure 13. Formalization of the arrangement task as case-based reasoning.



The second application example is a melodic similarity calculation for polyphony. Melodic similarity calculations play a central role in discovering a piece's structure with a music summarizer (Hirata and Matsuda 2002). To measure similarity between two polyphonic passages, we measure how much information is lacking from the two polyphonic passages by the *lub* calculation. As described before, *lub* calculates the largest common part or the most common information of two input passages. For instance, if the calculation result is equal to its input polyphonic passages, the *lub* calculation does not lose any information, and the input passages are considered identical (i.e., most similar). If the calculation result is, in contrast, ***T***, there is no common part, and they are considered unrelated (i.e., least similar). We think the lacking

information should be concerned with the time-span tree and the temporal structure, which correspond to the two aspects of music formalized in our framework.

To make the lacking information quantitative, we introduce three measures, R_N, R_A, and R_T, defined as follows.

$$R_{\$}(P,Q) = \frac{|lub(P,Q)|}{max(|P|_{\$},|Q|_{\$})}$$

where \$ represents N, A, and T. Here, let P and Q be polyphonic passages. $|P|_N$ represents the number of note objects in P, $|P|_A$ is the total number of attributes of all note objects in P, and $|P|_T$ is the total number of attributes of all *temp* objects in P.

In some sense, $|P|_N$ and $|P|_A$ quantify the information conveyed by the time-span tree, and $|P|_T$ quantifies the temporal structure. Therefore, R_N expresses the similarity of time-span trees at the note level, where an incomplete note is also regarded as a note. R_A expresses the similarity of time-span trees at the attribute level and indicates to what extent the pitch class, octave, duration, and onset-time attributes of all notes in the result of *lub* are instantiated. Likewise, R_T expresses the similarity of temporal structures at the attribute level, and it indicates to what extent the four attributes of temp objects for all notes in the result of *lub* are instantiated.

Let us examine the real values of these similarity measures. If $P = Q$, then $R_N = R_A = R_T = 1.0$, and if $lub(P,Q) = \boldsymbol{T}$ in contrast, then $R_N = R_A = R_T = 0.0$. For an interesting case, consider the similarity between P and Q such that $P \subseteq Q$. Then, because this leads to $|P|_{\$} \geq |Q|_{\$}$ for \$ = N, A, or T, we have $R_{\$} = |Q|_{\$}/|P|_{\$}$. For more details, please refer to Hirata and Matsuda (2002).

Discussion

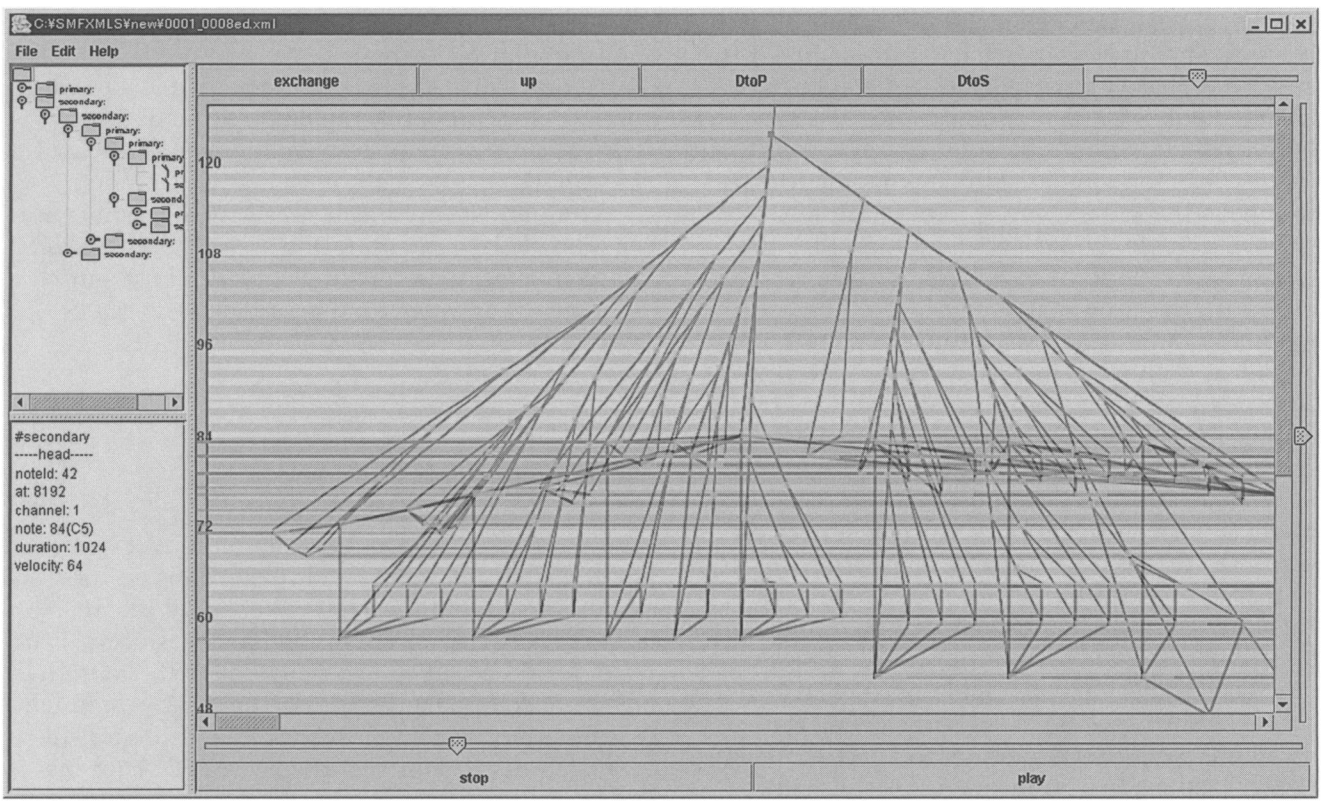
Related Work

Music representation methods and music description languages that do not employ GTTM include Balaban's Music Ontology (Balaban 1996), Dannenberg's Nyquist (Dannenberg 1997), and feature space (Rowe 1993; Cope 1996). Music Ontology is one of a few fundamental attempts toward a formal treatment of music. We think that Nyquist has the optimal (i.e., necessary and sufficient) specifications as a music description language. Cope's system is well known for its artistic compositions in various genres.

The models provided by Music Ontology and Nyquist use only the superficial relations of relevant notes on a score with their surrounding notes, which are essentially superimposition and juxtaposition. The operations on these superficial relations include get_next_note, interval_to_previous_note, and is_simultaneous, which work as the application program interface (API). The feature space method basically maps a given melody to a feature space, the dimensions of which typically include the number of notes occurring in a relevant measure, the average pitch of all notes, and the up/down vector of pitch intervals of a melody. These features are also regarded as superficial relations and selected in a more or less ad hoc way. These three approaches require more or fewer heuristics peculiar to each application, which imposes a different system model on the user, while they have an advantage in that their models are small and comprehensive.

Other attempts at employing GTTM include Widmer's performance-rendering system (Widmer 1995, 1996); Arcos's and de Mántaras's saxophone sound synthesizer (2000); and a learning system described by Uwabu, Katayose, and Inokuchi based on prolongational reduction from a corpus (1997). Unfortunately, Widmer, Arcos, and de Mántaras give just the schematic pictures related to knowledge representation. It does not seem that these data structures are suitably designed for the arguments of some mathematical operators. Uwabu and colleagues proposed an actual procedure for prolongational reduction by using the co-occurrence relations extracted from corpora, where a prolongational tree is described in the ML language, not a time-span tree. Their data structure is dedicated to discovering the co-occurrence relation, and the temporal information of a piece is not included.

Figure 14. Screen shot of the TS-editor currently under development, which shows the first seven bars of Mozart' "Rondo alla Turca" from Mozart's Sonata in A, K. 331.



Problems

We think there are four problems to be solved. First, GTTM claims that a head dominates the time span including it, and a head is hence placed at the beginning of a time span on a score (the "GTTM head"). It turns out that the notes during the time-span reduction do not represent real notes but rather the notes in the listener's recognition. Thus, Lerdahl and Jackendoff might not think that it was meaningful to examine the similarity between the original homophony and the intermediate melodies during the time-span reduction. Dibben (1994) investigated the similarity between an original homophony and the intermediate melodies with the GTTM head and reported an affirmative result, while there were no considerations on the idea of the conservative head. We would look forward to future investigations that compare the conservative head and the GTTM head from the listener's recognition point of view.

Second, our framework provides three primitive operators: the subsumption relation, least upper bound, and greatest lower bound. However, the descriptive power of these primitives may not be enough to express the functions for complicated musical tasks. Hence, we should invent or discover new primitive operators to gain more descriptive power.

Third, because music itself is comprised of many features, music should be analyzed from more than one aspect to obtain a more precise representation of musical intuition and concepts. This may lead to the use of other music theories, such as Preference Rule Systems, prolongational reduction of GTTM, and the implication-realization model. However, formalizing the music theories that lack the reduction concept will require some other techniques.

Additionally, efficiency of description and transposition should be improved. Because parts of *ts* and *temp* objects represent the same information (a stable note), we may improve their descriptions by introducing inheritance, for example. Next, our method cannot compare two polyphonic passages that have the same contour in different keys (i.e., transposed polyphony). We must formalize the transposition concept in our framework.

Fourth, the time-span analysis must be done manually in our framework, but we do not offer any ideas on how a time-span tree can be algorithmically derived from a musical surface in this article. This article focuses on music knowledge representation and presents usage of the time-span analysis results after computers automatically derive a time-span tree from a musical surface. We think at present it is the most appropriate for users to manually perform the time-span reduction of polyphony, not only because current technology to do so automatically is premature, but also because excessive automation may yield unintended analysis results. Hence, we are developing a dedicated editor (even for users who are not familiar with music theory), called TS-editor, that allows users to attach a time-span tree and temporal structures (see Figure 14).

A score is displayed in piano-roll format on the TS-editor's screen. The editor enables a user to efficiently indicate the relations among *head*, *primary*, and *secondary*, and the information of *stable* and *difference* in a temporal structure. For more details, see Hirata and Matsuda (2002).

Conclusion

This article proposed a music knowledge representation framework based on the generative theory of tonal music (GTTM). Our framework succeeds in formalizing polyphony to some extent from the GTTM point of view, though this work is merely a first step towards a methodology for designing intelligent music applications. To evaluate our framework, we will have to build several music applications and examine the descriptive power, precision, and efficiency of our framework.

Music theory has addressed musical intuition and recognition without the concept of automation for a long time. At present, there are a few ongoing projects for computer implementation of music theory, such as Temperley and Sleator (1999) and Lerdahl (2001). We believe that a fundamental problem for an intelligent music application is music knowledge representation as well as computer implementation of music theory. We hope our research will be a first step in bridging the gap between music theory and music knowledge representation.

Acknowledgments

We are very grateful to two anonymous reviewers for valuable comments and suggestions on the previous version of this article. We would like to thank Prof. Yuzuru Hiraga and Mr. Yoshihiro Takeuchi for their valuable suggestions to the material of this article. Thanks are also due to Dr. Shunichi Uchida, the director of Research Institute for Advanced Information Technology (AITEC), for his support.

References

Arcos, J. L., and R. L. de Mántaras. 2000. "Combining AI Techniques to Perform Expressive Music by Imitation." *AAAI Workshop: Artificial Intelligence and Music*. Menlo Park, California: AAAI Press, pp. 41–47.

Balaban, M. 1996. "The Music Structures Approach to Knowledge Representation for Music Processing." *Computer Music Journal* 20(2):96–111.

Botelho, M. 1994. "Tonal Grouping: An Addendum to Lerdahl and Jackendoff's 'A Generative Theory of Tonal Music'." In *Proceedings of the Third International Conference for Music Perception and Cognition*, pp. 265–266.

Carpenter, B. 1992. *The Logic of Typed Feature Structures*. Cambridge, UK: Cambridge University Press.

Castan, G., M. Good, and P. Roland. 1999. "Extensible Markup Language (XML) for Music Applications: An Introduction." *Computing in Musicology* 12:95–102.

Cope, D. 1996. *Experiments in Musical Intelligence*. Middleton, WI: A-R Editions.

Dannenberg, R. B. 1993. "Music Representation Issues, Techniques, and Systems." *Computer Music Journal* 17(3):20–30.
Dannenberg, R. B. 1997. "Machine Tongues XIX: Nyquist, a Language for Composition and Sound Synthesis." *Computer Music Journal* 21(3):50–60.
Dibben, N. 1994. "The Cognitive Reality of Hierarchic Structure in Tonal and Atonal Music." *Music Perception* 12(1):1–25.
Hirata, K., and T. Aoyagi. 1999. "A Musically Intelligent Agent for Composition and Interactive Performance." *Proceedings of the 1999 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 167–170.
Hirata, K., and T. Aoyagi. 2000. "Ba-Bi-Bun: Case-Based Reasoning Musical Arrangement System Understanding a User's Intention at Note Level." *IPSJ SIG Notes* 2000-MUS-37:17–23.
Hirata, K., and T. Aoyagi. 2001. "Pa-Pi-Pun: Creativity-Support Tool for Generating Jazz Harmony." *IPSJ Journal* 42(3):633–641.
Hirata, K., and R. Hiraga. 2000. "Next Generation Performance Rendering: Exploiting Controllability." In *Proceedings of the 2000 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 360–363.
Hirata, K., and S. Matsuda. 2002. "Interactive Music Summarization based on GTTM." In *Proceedings of the Third International Conference on Music Information Retrieval (ISMIR 2002)*. Paris: IRCAM, pp. 86–93.
Kifer, M., G. Lausen, and J. Wu. 1995. "Logical Foundations of Object-Oriented and Frame-Based Languages." State University of New York at Stony Book Department of Computer Science Technical Report TR-90–003.
Kolodner, J. 1993. *Case-Based Reasoning*. Burlington, MA: Morgan Kaufmann.
Lerdahl, F. 2001. *Tonal Pitch Space*. Oxford, UK: Oxford University Press.
Lerdahl, F., and R. Jackendoff. 1983. *A Generative Theory of Tonal Music*. Cambridge, Massachusetts: MIT Press.
London, J. 1994. "The 'Strong Reduction Hypothesis' of GTTM." In *Proceedings of the Third International Conference for Music Perception and Cognition*, pp. 261–262.
Narmour, E. 1990. *The Analysis and Cognition of Basic Melodic Structures – The Implication-Realization Model*. Chicago: University of Chicago Press.
Nord, T. A. 1992. "Toward Theoretical Verification: Developing a Computer Model of Lerdahl and Jackendoff's Generative Theory of Tonal Music." Ph.D. Diss., The University of Wisconsin–Madison.
Plaza, E. 1995. "Cases as Terms: A Feature Term Approach to the Structured Representation of Cases." *Lecture Notes in Artificial Intelligence*. Berlin: Springer-Verlag.
Rowe, R. 1993. *Interactive Music Systems: Machine Listening and Composing*. Cambridge, Massachusetts: MIT Press.
Sadie, S., ed. 1980. *The New Grove Dictionary of Music and Musicians*. London: Macmillan Publishers Ltd.
Selfridge-Field, E. 1997. *Beyond MIDI*. Cambridge, Massachusetts: MIT Press.
Temperley, D. 2001. *The Cognition of Basic Musical Structures*. Cambridge, Massachusetts: MIT Press.
Temperley, D., and D. Sleator. 1999. "Modeling Meter and Harmony: A Preference Rule Approach." *Computer Music Journal* 23(1):10–27.
Uwabu, Y., H. Katayose., and S. Inokuchi. 1997. "A Structural Analysis Tool for Expressive Performance." *Proceedings of the 1997 International Computer Music Conference*. San Francisco: International Computer Music Association, pp. 121–124.
Widmer, G. 1995. "Modeling the Rational Basis of Musical Expression." *Computer Music Journal* 19(2):76–96.
Widmer, G. 1996. "Learning Expressive Performance: The Structure-Level Approach." *Journal of New Music Research* 25:179–205.
Wiggins, G., E. Miranda, A. Smaill, and M. Harris. 1993. "A Framework for the Evaluation of Music Representation Systems." *Computer Music Journal* 17(3):31–42.
Yokota, K. 1992. "Towards an Integrated Knowledge-Base Management System: Overview of R&D on Databases and Knowledge Bases in the FGCS Project." *Proceedings of International Conference on Fifth Generation Computer Systems*. Institute for New Generation Computer Technology, pp. 89–112.