Melodic Morphing Algorithm in Formalism

Keiji Hirata[1], Satoshi Tojo[2], and Masatoshi Hamanaka[3]

[1] NTT/Future University Hakodate
[2] Japan Advanced Iinstitute of Science and Technology
[3] PREST, JST/University of Tsukuba

Abstract. We introduce a feature structure, corresponding to a time-span tree based on 'A Generative Theory of Tonal Music' (GTTM) for a music piece, and represent the reduction of the tree by the subsumption among these feature structures. As the collection of them forms a lattice, we can define the *join* and *meet* operations. We show a melodic morphing algorithm based on these simple operations.

1 Introduction

To facilitate composing music, we often render a pitch event, a chord, and so on in a formal representation, together with supporting tools. However, there exists a trade-off between descriptive power and simplicity in the formal representation, two of which are basically incompatible. Descriptive power is the capability as to how faithfully the composer's original thoughts and emotions are expressed. On the other hand, simplicity means how concise the description of the music is. In general, the more abstract a description is, the shorter it is, and the less the product of writing and reading costs is. For instance, a Standard MIDI File has high descriptive power yet low simplicity, while the chord symbol is the opposite. The trade-off can also be considered an issue of controllability in rendering music.

We argue that the key for being compatible is to separate the basic well-understood operations from creator's intention for combining them. Intuitively, these basic operations include the ones like set arithmetics; addition, subtraction, intersection, and union. Thus we are led to an algebraic framework, in which a creator assembles basic operations into a calculation process for a target task as the creator intends.

The aim of the paper is to prepare the theoretical foundations for proving the theorem for the property of a complicated musical task, melodic morphing. We start with defining the subsumption relation among melodies and building a lattice of feature structures, each element of which corresponds to a melody. Using the *join* and *meet* operations in the lattice, we construct a melodic morphing algorithm in the formal way.

2 Time-Span Trees in Feature Structures

First, we design the feature structure, *f-structure* hereafter, for a time-span tree of a music piece. An f-structure is a directed acyclic graph as used in [**?**]. Since

$$\begin{bmatrix} \tilde{}tree \\ head\ \{\{\boxed{\text{i}}, \boxed{\text{j}}\}, [\tilde{}event]\} \\ dtrs\ \{ \begin{bmatrix} left & \begin{bmatrix} \tilde{}tree \\ head\ \boxed{\text{i}}\ [\tilde{}event] \\ dtrs\ \{\cdots\} \end{bmatrix} \\ right & \begin{bmatrix} \tilde{}tree \\ head\ \boxed{\text{j}}\ [\tilde{}event] \\ dtrs\ \{\cdots\} \end{bmatrix} \end{bmatrix}, \bot \} \end{bmatrix} \qquad \begin{bmatrix} \tilde{}event \\ pitch & Pitch \\ pos & \begin{bmatrix} bar & Integer \\ meter & Length \end{bmatrix} \\ duration & Length \end{bmatrix}$$

Fig. 1. Feature structures for a time-span tree (left) and a pitch event (right)

the reduction is an intrinsic method to reflect musical structures [**?**], we can properly map the reduction of the time-span trees to the subsumption relation '$\sqsubseteq$' between f-structures.

The type of an f-structure is shown headed by '~' (tilde). An f-structure for a $\tilde{}tree$ is shown on the left-hand side of Fig. **??**, and that for an $\tilde{}event$ on the right-hand side. A binary tree has left and right branches where there are trees recursively; we call these branches daughters (*dtrs*). Let σ be a $\tilde{}tree$ f-structure, then the left daughter and the right daughter, denoted by $\sigma.dtrs.left$ and $\sigma.dtrs.right$ respectively, have recursively $\tilde{}tree$ type. The set notation $\{x, y\}$ means the choice either of x or y. When $\sigma.dtrs$ is $\bot$ (empty), that is, there is no daughter in the tree, $\sigma.head$ must be a single pitch event.

The whole f-structure is referred to by a tag, which is shown by such a boxed number as $\boxed{\text{i}}$ or $\boxed{\text{j}}$. The head of a $\tilde{}tree$ f-structure must be either the head of its left daughter tagged by $\boxed{\text{i}}$, that of its right daughter tagged by $\boxed{\text{j}}$, or another single pitch event. If $\sigma.head = \sigma.dtrs.left.head$, the node has the right-hand elaboration of shape $\diagup\!\diagdown$, and if $\sigma.head = \sigma.dtrs.right.head$, the left-hand elaboration $\diagdown\!\diagup$. As such, the *head* value at the parent level is recursively taken from either the left-hand or right-hand branch. As for $\tilde{}event$ type, feature *pitch* include $C4$, $B\flat 6$, $F\sharp 3$, and so on. Feature *pos* stands for the start timing, and its value is an f-structure consists of feature *bar* (the n-th bar) and *meter* (m-th meter measured by a quarter note). Feature *duration* also has a *Length* value.

A type of an f-structure specifies a set of indispensable features. When no indispensable feature is missing, the typed f-structure is said to be *full-fledged*. For example, $\tilde{}tree$ type requires the feature set of *head* and *dtrs*, and $\tilde{}event$ type does *pitch*, *pos.bar*, *pos.meter*, and *duration*. The property 'full-fledged' is concerned with whether or not rendering a real melody from a f-structure representation. Then, we can provide the formal definition of subsumption relation between f-structures, which allows the mechanical calculation.

3 Calculus in Melody Lattice

Theorem 3.13 in Carpenter [**?**] defines that the unification of f-structures A and B is the least upper bound of A and B. Therefore, we adopt the unification as the

definition of *join* in this paper, and the intersection of the unifiable f-structures as that of *meet*.

Definition 1 (*Meet and Join*). *Let A and B be full-fledged f-structures representing the time-span trees of melodies A and B, respectively.*

If we can fix the greatest lower bound of A and B, that is, the greatest x such that $x \sqsubseteq A$ and $x \sqsubseteq B$ is unique, we call such x the meet of A and B, denoted as $A \sqcap B$.

If we can fix the least upper bound of A and B, that is, the least y such that $A \sqsubseteq y$ and $B \sqsubseteq y$ is unique, we call such y the join of A and B, denoted as $A \sqcup B$.

Definition 2 (Reduction Path). *Suppose that each pitch event is given a beat strength as a result of metrical analysis [**?**]. For such two f-structures (melodies) A and B that $T_B \sqsubseteq T_A$, a reduction path from A to B is defined as a sequence of f-structures (melodies) obtained by removing a pitch event (a note) in $A \setminus B$ from A one-by-one, according to the algorithm, as follows:*

Step 1: $N := \#(T_A \setminus T_B)$, $i := 0$, and $T_0 := T_A$.

Step 2: Select a pitch event p of the minimum beat strength in $T_i \setminus T_B$.

Step 3: Reduce T_i to T_{i+1} by removing p.

Step 4: Iterate Steps 2 and 3 N times ($i = 0 \sim N - 1$).

The resulting sequence T_0, T_1, T_2, $\cdots$, T_N is the reduction path from A to B.

Note that at Step 2 the pitch events (notes) with the minimum beat strength are the least important notes in the time-span tree. Since such least important notes are multiple and one is chosen from them nondeterministically, there are more than one reduction path in general. The selection is justified by the time-span reduction preference rules: TSRPR1 (metrical position) and TSRPR5 (metrical stability) [**?**]. Hence the algorithm may automatically compute more than one reduction paths, for every melody C on the reduction path from A to B, $B \sqsubsetneq C \sqsubsetneq A$ holds.

4 Melodic Morphing Algorithm

Here, we present the morphing algorithm in the formal way[**?**,**?**].

Definition 3 (Melodic Morphing Algorithm). *The algorithm consists of the following steps (Figure **??**):*

Step 1: Calculate $T_A \sqcap T_B$ (meet).

Step 2: Select melody T_C on the reduction path from T_A to $T_A \sqcap T_B$, and select T_D on the reduction path from T_B to $T_A \sqcap T_B$.

Step 3: Calculate $T_C \sqcup T_D$ (join), and the result is morphing melody μ.

If C close to T_A is chosen, the characters of melody A are reflected in output μ. On the other hand, If C close to $T_A \sqcap T_B$ is chosen, those of A are less emphasized in μ. The character of D, as opposed to B, behaves in the similar way.

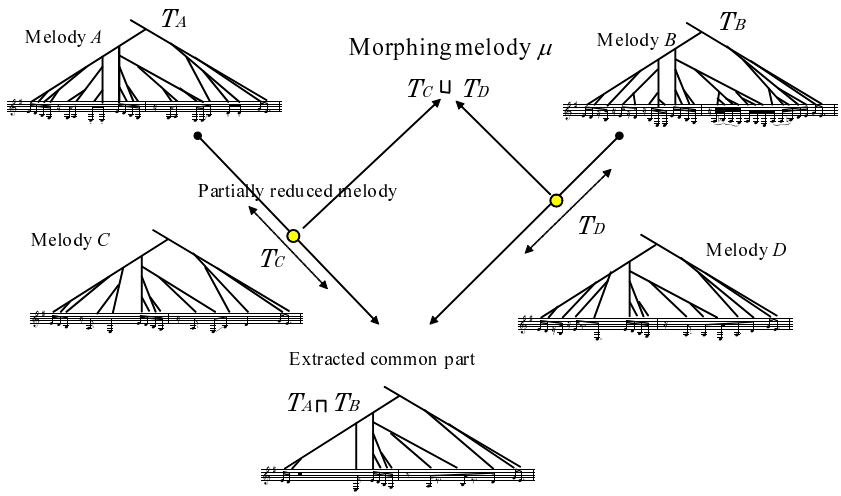

Fig. 2. Melodic morphing algorithm

5 Conclusion

We have provided an algebraic framework in which a music piece is represented by an f-structure, corresponding to its time-span tree. As these f-structures were ordered in terms of the subsumption relation, we could construct a lattice of melodies and could define *join* and *meet* operations. We have presented a morphing algorithm, which is a complicated calculation in general, from combinations of these simple algebraic operations. In the near future, we will prove the algorithm is exactly the interpolation of two given melodies.

References

1. Carpenter, B.: The Logic of Typed Feature Structures. Cambridge University Press (1992)
2. Hamanaka, M., Hirata, K., Tojo, S.: Melody Morphing Method Based on GTTM. In: Proc. of ICMC 2008, pp.155–158 (2008)
3. Hamanaka, M., Hirata, K., Tojo, S.: Melody Extrapolation in GTTM Approach. In: ICMC 2009, pp. 89–92 (2009)
4. Lerdahl, F., Jackendoff, R.: A Generative Theory of Tonal Music. The MIT Press, Cambridge (1983)
5. Marsden A.: Generative Structural Representation of Tonal Music. J. New Music Research, vol. 34, no. 4, pp. 409–428 (2005)
6. Selfridge-Field, E.: Conceptual and representational issues in melodic comparison. Computing in Musicology vol. 11, pp. 3–64, The MIT Press, Cambridge (1998)