Interactive Music Summarization based on GTTM

Keiji Hirata
NTT Communication Science Laboratories
3-1 Morinosato Wakamiya, Atsugi-shi
Kanagawa 243-0198, Japan
+81-46-240-3658
hirata@brl.ntt.co.jp

Shu Matsuda
Digital Art Creation
2-9-1-506 Fuchucho, Fuchu-shi
Tokyo 183-0055, Japan
+81-42-361-2427
shu@dacreation.com

ABSTRACT

This paper presents a music summarization system called "Papipuun" that we are developing. Papipuun performs quick listening in a manner similar to a stylus skipping on a scratched record, but the skipping occurs correctly at punctuations of musical phrases, not arbitrarily. First, we developed a method for representing polyphony based on time-span reduction in the generative theory of tonal music (GTTM) and the deductive object-oriented database (DOOD). The operation, least upper bound, plays an important role in similarity checking of polyphonies represented in our method. Next, in a preprocessing phase, a user analyzes a set piece by the time-span reduction, using a dedicated tool called TS-Editor. For the real-time phase, the user interacts with the main system, Summarizer, to perform music summarization. Summarizer discovers a piece structure by means of similarity checking. When the user identifies the fragments to be skipped, Summarizer deletes them and concatenates the rest. Papipuun can produce a music summarization of good quality, reflecting the atmosphere of an entire piece through interaction with the user.

1. INTRODUCTION

Music summarization is a significant and attractive task because it potentially can reconstruct the architectures of music systems and, in addition, open up many new applications. Let us consider standard musical tasks, such as composition, arrangement and performance. Many conventional music systems have been developed for these tasks. Since these tasks are coarse-grained and highlevel, non-experts in music likely regard them as a black box full of profound unknown experience, skill, knowledge, and talent. However, we believe that these tasks can be decomposed to finer-grained lower-level tasks, such as music summarization and music information retrieval. If a music system provides a non-expert user with middle- to low-level tasks, such as summarization and retrieval, the user may be able to just combine them to design a high-level task without having to take musical trifles into account. We think that introducing middleware-level tasks will make it easier for a non-experts in music to use a music system. Such a framework will therefore facilitate the tendency toward making musical technologies more oriented to non-expert end-users.

Recently, Internet and Web technologies have become widespread, and various kinds of knowhow and information have been distributed and accumulated. The combination of middleware-level tasks, such as music summarization and retrieval, with Internet/Web technologies will therefore lead to new applications, such as intelligent karaoke, ringing tones of mobilephone, and interactive musical art/business on Internet. Some are potentially killer applications.

The task of music summarization is to find the most distinctive or representative musical excerpts (automatically). There have been a few approaches to music summarization. They can be split into audio-signal and symbolic approaches. The former includes that by Logan and Chu [8], who asserted that the most interesting and memorable part of the song, called a key phrase, is that which occurs most frequently. However, this is not always true because a key phrase usually depends on a user's preference and taste. In addition, Logan and Chu's method cannot generate a summarization reflecting the atmosphere of an entire piece. The symbolic approach is focused on a symbolic representation of music, which corresponds to the description level of a score, SMF, and so on. We think that the symbolic approach facilitates the use of music theory, more than the audio-signal approach. Huron's approach [5] is a symbolic one, though the details are not described in his paper unfortunately.

We aim at achieving music summarization that reflects the atmosphere of an entire piece with a symbolic approach. For this purpose, it is important to consider that an intelligent summarization requires musical knowledge and skills provided by music theory to analyze a complicated structure of a piece.

This paper presents a prototype system that we are developing named "Papipuun". Papipuun performs quick listening by stylus skipping, where the skipping occurs correctly at punctuations of musical phrases, not at arbitrary positions. The current version of Papipuun can accept just piano pieces, such as "Turkish March" by Mozart or "Let It Be" by the Beatles, arranged for solo piano (i.e. the format 0 SMF of a single instrument). We think that this restriction is a good starting point for future enhancement because it highlights the essential issues to be solved, which are music representation of polyphony, discovery of a piece's structure by similarity checking, identification of fragments to be skipped, and concatenation of selected fragments.

This paper is organized as follows. We introduce a method of representing polyphony in Section 2, present the similarity checking technique based on operation, least upper bound, in Section 3, and propose a method of interactive music summarization and describe Papipuun in Section 4. Lastly, we make concluding remarks and mention future work in Section 5.

2. MUSIC REPRESENTATION

The purpose of designing our music representation method is to describe the relationships between polyphonies with respect to time-span reduction and to use time-span reduction for similarity checking of polyphonies. We have designed a data structure for representing polyphony based on time-span reduction in the generative theory of tonal music (GTTM) [7] and the deductive object-oriented database (DOOD) [1] [11, 6]. A reharmonizer [2] and a performance rendering system [4] were also developed in the framework of GTTM and DOOD.

2.1 Brief Introduction to GTTM and DOOD

Lerdahl and Jackendoff (1983) proposed GTTM as a theory for formalizing the listener's intuition. GTTM provides concepts and procedures for analyzing and interpreting Western tonal music written in common music notation, namely a score. A score is just a surface structure of music, and GTTM derives from the surface structure the score's hidden hierarchical structures, which make up the underlying deep structures. Note that the music treated by GTTM is limited to homophony. It is said that GTTM is modeled on Chomsky's transformational grammar since it employs the framework of the surface and deep structures. Among the many theories proposed for analyzing music, we consider GTTM the most appropriate for computer implementation since it is the most formally constructed one, although even more efforts at formalizing GTTM are needed before we can write a working program.

GTTM comprises a grouping structure analysis, a metrical structure analysis, time-span reduction, and prolongational reduction. The grouping structure analysis segments a homophony into shorter groups. It seems that when we sing a long melody, we have to find the proper points for drawing a breath. The metrical structure analysis identifies a series of tacti and beats of a homophony and the positions of strong (or weak) beats at the levels of a quarter note, half note, a measure, and so on. It seems that a conductor moves a baton to the music played or a listener keeps time by hand clapping. In particular, time-span reduction represents the intuitive idea that if we remove grace notes from a long melody, we obtain a simple similar sounding melody. An entire piece of music can eventually be reduced to a key 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. We do not describe prolongational reduction here because of space limitation.

DOOD is a knowledge representation method with a theoretical foundation and is thus tractable. DOOD is an extension of the first order logic in that it can represent the lack of attributes and type declaration [11]. From the knowledge description point of view, DOOD has almost the same functions as the feature structure [1]. Plaza (1995) claimed that the feature structure is suitable for describing a case (including melody). A prominent feature of DOOD is that a deductive rule for DOOD terms to formally define any relation is available; this yields descriptive efficiency, accuracy, and expandability.

The DOOD framework is motivated by the introspection that things in the real world can be represented as the combination of a basic (atomic) object and a set of the 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 to stand for a basic object, $l : v$ an attribute, l an attribute name (label), and v an attribute value. The most fundamental relation in the real world 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 special object $\subseteq$ a generic object".

We assert that the counterpart of the subsumption relation in GTTM is the time-span reduction [3]. We think that this correspondence is the most natural.

2.2 Abstracting and Instantiating Melody using Time-Span Reduction

A time-span tree in GTTM is a binary tree. In this article, we refer to an important branch as primary and the other as secondary. The left-hand side of Figure 1 depicts a simple melody and its time span tree. The time span (designated as ◂·····▸) covered by a primary and secondary branches is represented by a single note, called a head, which is here designated as "C4".

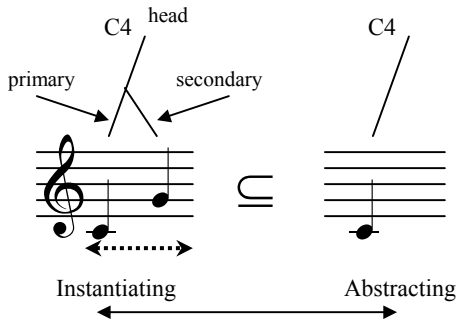

Figure 1. Subsumption relation of melodies.

Our representation method can represent such a time-span tree and the temporal information of every note of a melody, although the temporal information is not explicitly showed here. The temporal information is needed in order to perform the time-span reduction correctly and includes the temporal relationships between a relevant note and its surrounding notes as well as the absolute onset timing and the duration of each note.

The melody on the right-hand side of Figure 1, consisting of only the C4 note, can be considered more abstract than that on the left-hand side from a time-span reduction point of view. In other words, the melody on the left-hand side is more instantiated than that on the right-hand side. This abstraction-instantiation relation of the melodies is regarded as a kind of the subsumption relation (the partial order) and is thus written "$\subseteq$". Note that when we describe these two melodies using our music representation method, it can be mechanically derived that this subsumption relation holds by looking at the actual data structures of the melodies. In a strict interpretation of the time-span reduction in GTTM, the right-hand side should be a half note of C4. However, giving priority to mechanization, we determine that the notes remaining after the time-span reduction have the same onset timings and durations as before.

The shape of a time-span tree, the head values, and the temporal structure depend on the interpretation (prior analysis) of a melody. Once an interpretation is given and it is represented as a data structure in our method, the data structure conforms to the subsumption relation in DOOD. If a different interpretation of a melody is supposed, then the shapes of the time-span tree, the head values, and the temporal information are different, and different subsumption relations hold.

Let us show another example of Alberti bass in Figure 2. Reference [7] lists four ways of generating a head value. In Figure 2, the head values are generated by "fusion", while "ordinary reduction" in Figure 1. The way of generating a head value also depends on the interpretation of a melody given for analysis.

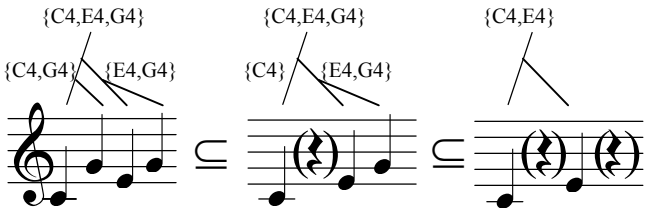

Figure 2. Subsumption relations of Alberti bass

Since our representation method preserves the onset time and the duration of a relevant note upon time-span reduction, when a less important note is removed, the time span occupied by the note is substituted with an interval without a note. As a result, the interval can be regarded as a rest with the duration of the removed note. Thus, the rest marks in the figure are put in parentheses. They may correspond to the rests in SMF.

Hereafter, for space efficiency, we omit the G clef.

2.3 Reduction to Ordering Information

The temporal structure of our representation method was conceptualized by straightforwardly intuiting that abstracting the difference between onset times should produce the ordering information between them, not absolute timings.

Figure 3 shows examples where the two melodies have the same pitch sequence but different onset timings and durations.

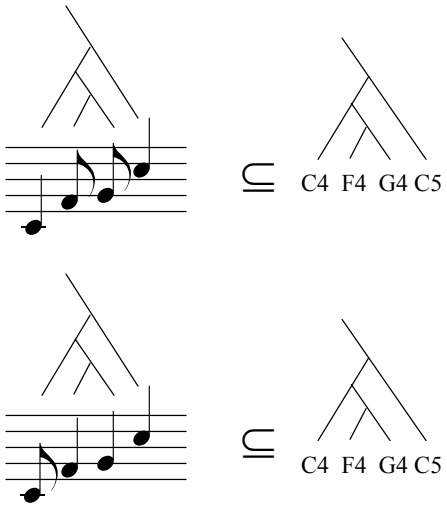

Figure 3. Reduction of different melodies of same pitch sequences to identical pitch ordering.

Upon listening to these two melodies, one may recognize something common to them: they consist of the same pitches, whose orderings are identical, i.e. “C4 F4 G4 C5” in the figure. This observation implies that reducing a melody should yield a note sequence consisting of the same pitches with just the ordering information preserved. Note that the original melodies and the abstract note sequence (temporally reduced melody) share the same time-span tree. Moreover, our method can represent a melody that is partly reduced to the ordering and determine the subsumption relation between it and some other abstract or instantiated melody as well.

Since the subsumption relation in Figure 3 holds in our representation method, we think that our method succeeds in formalizing the tacit assumption for the temporal aspect of the time-span reduction.

2.4 Representing Polyphony

The scope of GTTM is limited to just homophony for theoretical reasons. But for high applicability, we think that a practical music system should treat polyphony, taking into account its deep structures. Thus, we extend the time-span reduction so that it can treat polyphony. Basically, polyphony means a texture achieved by the interweaving of several melodic lines that are independent but work together harmonically. For our purpose, we need the following more formal definition of polyphony. First, a homophony is defined as a melody superimposed by subordinate melodies and interpreted as a single melody temporally. Next, a polyphony is defined inductively. A homophony is a polyphony, and a node of a time-span tree consisting of two polyphonies is a polyphony, where these two polyphonies may be temporally overlapped. That is, the principle of our method is to regard two possibly overlapped polyphonies just as two branches of a time-span tree’s node by coercively making the order between them with respect to the time-span reduction.

Figure 4 shows an example of polyphony and our extended time-span reduction.

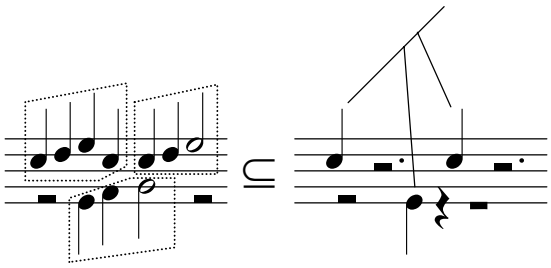

Figure 4. Time-span reduction of polyphony.

On the left-hand side of the figure, each polygon of dotted lines shows a homophony obtained by decomposing the original polyphony into components until they become homophonies. On the right-hand side, only the heads are left with the same onset timings and durations. Note that the decomposition into homophonies and the time-span tree in the figure is one way of analyzing the sample polyphony. If a different analysis result is given, another decomposition and time-span tree are constructed.

3. SIMILARITY CHECKING

We use the subsumption relation to construct an algorithm for checking the similarity between two polyphonies, where the core operation is least upper bound (*lub)*. Intuitively, the operation *lub* calculates the largest common part of two given melodies. We usually define operation *lub(x, y)* as $min(\{z \mid x \subseteq z \wedge y \subseteq z\})$. For all x and y, $x \subseteq lub(x,y)$ and $y \subseteq lub(x,y)$. The mathematical meaning of *lub* is well known. If two completely different melodies that share no common part are given, the calculation result is a vacancy, which means the most abstract, least informative melody, denoted as $\top$ (called top). Since $\top$ corresponds to a vacancy, we can not always listen to the calculation result of *lub*. For instance, if a note in the calculation result of *lub* has timing and duration sufficiently instantiated yet $\top$

for pitch, the note cannot be translated to an actual note on a score because its pitch is not determined.

In our method, the similarity checking between two melodies is based on *lub*. Since calculating *lub* of two melodies results in their largest common part, the larger the common part is, the more similar to each other the two melodies are. Plaza [9] claimed that the feature structure [1] is suitable for describing cases and also used *lub* for finding cases most similar to a query, although the concept of Plaza's representation method for music is essentially different from ours.

3.1 Similarity based on *lub*

Figure 5 shows an example of calculating *lub* of two simple melodies, C4 G4 and C4. The time-span trees of the melodies are shown, whereas their temporal structures are omitted for simplicity. For this example, since subsumption relation "melody C4 G4 $\subseteq$ melody C4" holds, the result is melody C4.

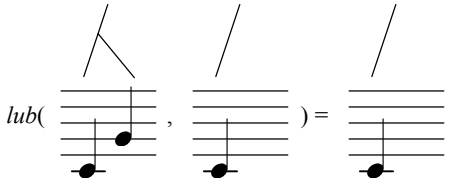

Figure 5. Simple example of *lub*.

Figure 6 shows examples of more complicated melodies. The middle notes of the two melodies are D4 and F4.

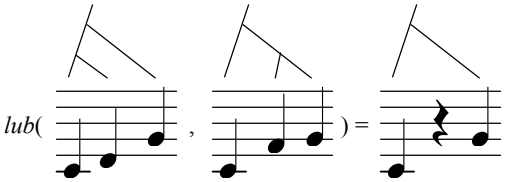

Figure 6. Largest common part by *lub*.

Since the notes do not match each other in terms of pitch and the time-span tree (accordingly, temporal structure, as well), the resulting melody does not include the middle note.

Figure 7 shows the same sample melodies as in Figure 3, where the two given melodies having the same pitch sequence yet different durations.

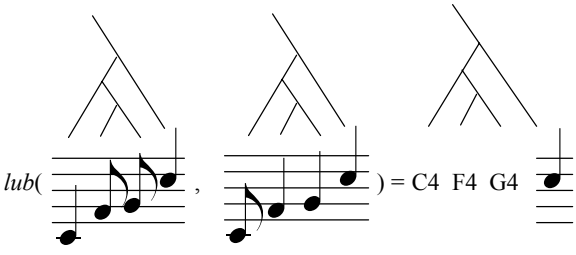

Figure 7. Abstract notes and ordering information.

The result of *lub* of the two melodies is that notes C4, F4, and G4, with their durations unknown, form a sequence in this order that ends with a quarter note of C5. That is, the resulting abstract melody is the most informative one that can be obtained from the two input melodies. Notes C4, F4, and G4 are considered incomplete in the sense that their pitches are determined but their durations are not.

Let us suppose that two melodies, D4 C4 and C4 G4, both have note C4 at the origin time (Figure 8). Due to the temporal structure of our method, our method can align the input melodies and then perform *lub*. In Figure 8, the alignment is shown by the bar lines immediately followed by C4.

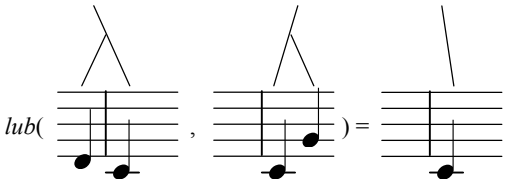

Figure 8. Temporal alignment.

The temporal structure introduced enables us to treat a melody starting at auftakt.

In contrast, Figure 9 shows the result of *lub* of the same two melodies as in the previous example, D4 C4 and C4 G4. Since the notes at the origin time are different however, the result is *T*.

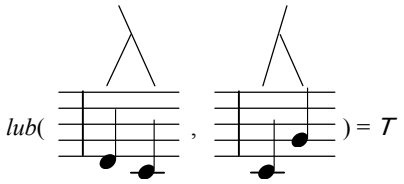

Figure 9. *lub* of incoherent melodies.

All examples above treat only monophonies for ease of explanation, but the *lub* of our method can treat polyphony as well.

3.2 Similarity Measures

Intuitively speaking, since *lub* calculates the largest common part of two input polyphonies, if the calculation result is equal to either of the two polyphonies, there is no loss of information by the calculation of *lub*, and the two polyphonies are considered identical (most similar). If the calculation result is, in contrast, *T*, there is no common part, and they are considered unrelated (least similar). Thus, our concept for measuring similarity between two polyphonies is to measure how much information in the *lub* calculation is lacking from the two polyphonies.

The lacking information is concerned with the time-span tree and the temporal structure, which correspond to the aspects of music formalized by our method. To express similarity parameters, we here introduce mathematical notations. Let P be a polyphony. Then, $|P|_N$ means the number of notes in P, $|P|_A$ the total number of attributes of all note objects in P, and $|P|_T$ the total number of attributes of all timing objects in P. Since a single note object has two attributes, pitch/duration and timing, $|P|_A = |P|_N * 2$ for well-formed P. Similarly, since a single timing object has four

attributes, i.e. predecessor, successor, salient note, and difference, $|P|_T = |P|_N * 4$ for well-formed P. Then we introduce three measures, R_N, R_A, and R_T, to make the lacking information quantitative. Let P and Q be polyphonies. Then

$$R_\$(P,Q) = \frac{|lub(P,Q)|_\$}{max(|P|_\$,|Q|_\$)} \qquad \$ \text{ is N, A and T, respectively}$$

R_N and R_A are associated with the time-span tree, and R_T the temporal structure.

R_N is the similarity of time-span trees at the note level, where an incomplete note is also regarded as a note. R_A is the similarity of time-span trees at the attribute level and indicates to what extent the pitch and duration attributes of all notes in the result of *lub* are instantiated. Likewise, R_T is the similarity of temporal structures at the attribute level and indicates to what extent the four attributes of all notes in the result of *lub* are instantiated. Note that all the attributes of notes and of temporal structures are assumed to be equally weighted.

Let us look over real values of these similarity measures. If $P = Q$, $R_N = R_A = R_T = 1.0$, and if $lub(P,Q) = T$ in contrast, $R_N = R_A = R_T = 0.0$. For an interesting case, consider the similarity between P and Q such that $P \subseteq Q$. Then since this leads to $|P|_\$ \leq |Q|_\$$ for \$ = N, A, or T, we have $R_\$ = |P|_\$ / |Q|_\$$. For the sample in Figure 6, we obtain $R_N = 2/3$, $R_A = 2/3$, and $R_T = 5/9$. For the sample in Figure 7, we obtain $R_N = 1.0$, $R_A = 5/8$, and $R_T = 10/13$. For the sample in Figure 8, we obtain $R_N = 1/2$, $R_A = 1/2$, and $R_T = 1/5$. Here we do not explain in depth how these values are calculated because of space limitation, and please understand that these values just give a feeling of the multiple viewpoints of polyphonic similarity.

Conventional working approaches for representing and comparing music mostly focus on surface information, not deep structures [10]. A relevant way of taking into account deep structures is to adopt the concept of reduction. Selfridge-Field (1998) claimed that the reductions suited for melodic searching and comparison may not be as concise as those used in the implication-realization model by Narmour, which was developed for analytic purposes, although she did not mention the time-span reduction in GTTM. We think that our method proposed here is a practical answer to her claim.

4. INTERACTIVE MUSIC SUMMARIZATION

As described in Section 1, a music summarization proceeds in these steps: (1) discovery of piece structure by similarity checking, (2) identification of fragments to be chosen or skipped, and (3) concatenation of chosen fragments. The operations of our prototype system, Papipuun, are designed in accordance with these steps (Figure 10).

In the preprocessing phase, a user analyzes a set piece in SMF using the time-span reduction and generates the information of corresponding time-span trees using a dedicated tool, TS-Editor. In the real-time phase, the user interacts with a main system, Summarizer, to perform music summarization (Section 4.3). Summarizer first looks for similar fragments in the piece using the *lub* of time-span trees of the fragments, groups them, and displays them on its window (Figure 15). Several parameters are given for changing the behavior of the polyphonic similarity, some of which are used as threshold values of R_N, R_A, and R_T. A user repeatedly changes the similarity parameters and examines the intermediate results of grouping similar fragments using Summarizer's GUI.

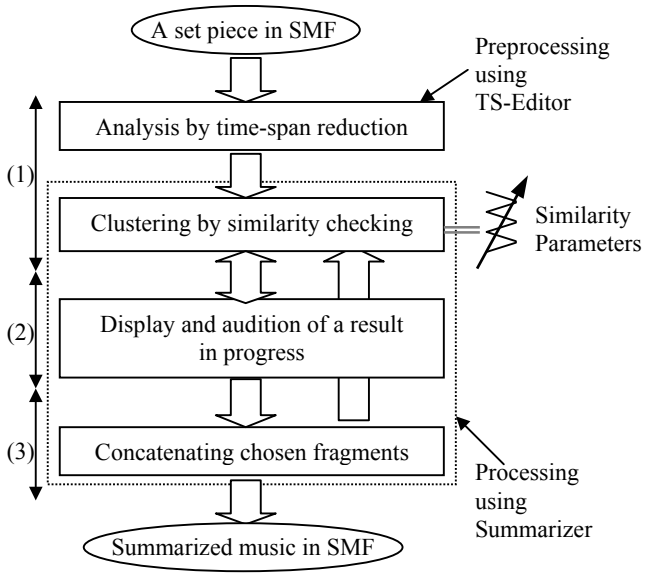

Figure 10. System operations of Papipuun

When the fragments to be skipped (opposite of left) are fixed, Summarizer deletes them, concatenates the rest, and lets a user listen to the resulting concatenation. If the concatenation is acceptable to the user, the session is terminated, and otherwise the user may return to the step of changing the similarity parameters. TS-Editor and Summarizer are both implemented in Java.

Our system design policy here is that the modules underlain by the music theory operate automatically and ones not underlain by the music theory are operated manually. In this case, the method for representing polyphony is underlain by GTTM, but a method for music summarization is not underlain by any music theory. Thus, all modules up to and including clustering of similar fragments are done automatically, and modules after this, manually. To automatize the manual modules, such as choosing fragments to be concatenated, various heuristics may be applied to them. However, the resulting concatenation is not always preferable and may be rather frustrating. For the present, we think that our system design policy is proper.

4.1 Preprocessing by TS-Editor

TS-Editor is a dedicated tool that allows a user can enter the time-span tree of a set piece. The input is the SMF file of a set piece, and the output is two XML files that are the translation of the input SMF file and the data of the corresponding time-span tree.

Figure 11 shows the window of TS-Editor at work. It displays a polyphony in the piano roll format and its corresponding time-span tree (the first seven bars of "Turkish March"). The subwindow at the upper left also displays the same time-span tree in the folder format for ease of pointing at intermediate nodes. The input procedure has two steps: the time-span tree and temporal structure.

First, when a SMF file is read into TS-Editor, all notes are sorted chronologically and a default time-span tree is provisionally attached to them such that a relevant note is a left-branching elaboration of its chronological successor (Figure 12a).

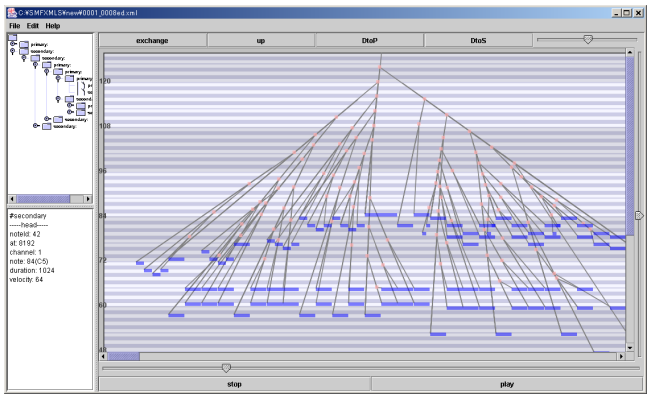

Figure 11. Polyphony in the piano roll fomat and its time-span tree on the window of TS-Editor.

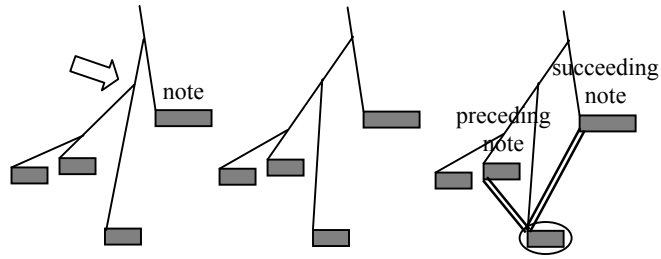

Figure 12 Editing a time-span tree and attaching a temporal structure.

In the figure, each gray box stands for a note of a piano roll score. Suppose a user wants to obtain the time-span tree in Figure 12b from that in Figure 12a. A user selects the node designated by the arrow in Figure 12a and issues the exchange command. TS-Editor provides four operators including the exchange command to manipulate a time-span tree. Also, TS-Editor gives a user a command to group more than one note as a chord.

After entering a time-span tree, the user inputs the temporal structure information. In Figure 12c, we assume that a relevant note, enclosed by an oval, occurs between preceding and succeeding notes. To express this temporal relation, a user attaches two double lines, one between the relevant and the preceding notes and the other between the relevant and the succeeding notes.

Figure 13 shows the same part as Figure 11 when a user finishes attaching the temporal structure to every note included in the piece[2].

TS-Editor greatly improves the efficiency of entering a time-span tree and temporal structures. In the case of entering "Turkish March" using the current version of TS-Editor, it takes about three hours for four bars on average, where a bar approximately contains 12.9 notes on average.

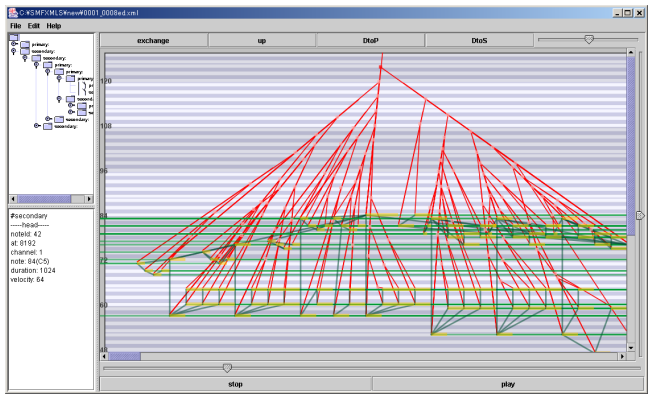

Figure 13. Polyphony with its time-span tree and its temporal information.

4.2 Clustering of Similar Fragments

Figure 14 is a startup snapshot of Summarizer's GUI taken just after loading the output files of TS-Editor. The entire score of "Turkish March" is also displayed in the piano roll format. According to the standard notation, a piece structure would be expressed as AABA or ABA'. However, the GUI instead shows a piece structure by putting on a piano roll score colored rectangular slices for indicating similar fragments.

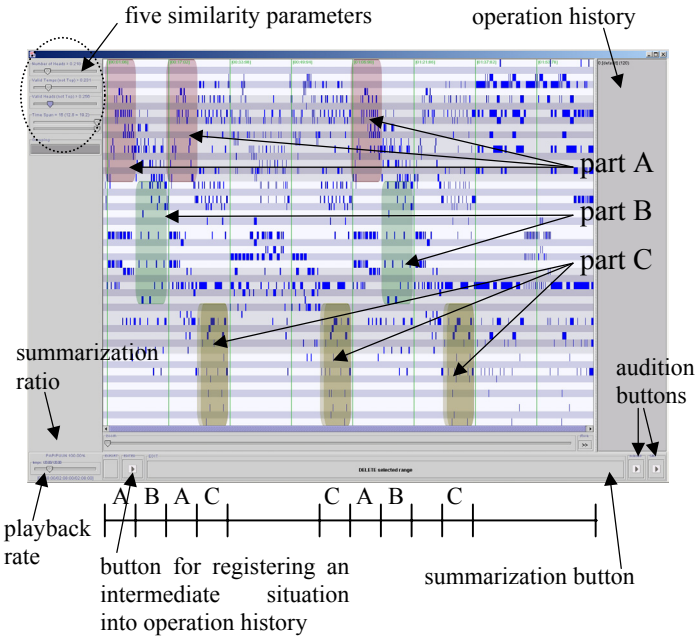

Figure 14. Startup snapshot of Summarizer's GUI.

At the upper left, there are five sliders for controlling the similarity parameters. The first three from the top are the threshold values for R_N, R_T, and R_A, called T_N, T_T, and T_A, respectively ($T_\$$ = 0.0~1.0 for \$ = N, T, or A). The next two are fragment size S and a margin for fragment size M. Within Summarizer, a fragment of a piece is represented as a corresponding subtree of the time-span tree of the entire piece.

Given two fragments of a piece, P and Q, Summarizer determines that P and Q are similar to each other if $R_M(P,Q) > T_N \wedge R_T(P,Q) > T_T \wedge R_A(P,Q) > T_A$. Fragment size S can be varied 2, 4, 8 or 16 beats, and a margin for fragment size M can be varied from 0% to 20%. For instance, when $S = 4$ and $M = 10\%$, the time spans of all the fragments for calculating the similarity measures are restricted within 3.6 (= 4 * 0.9) to 4.4 (= 4 * 1.1) beats. Here, the length of one beat is obtained from the Tempo meta event in an input SMF file.

When these similarity parameters are fixed, Summarizer pairwise calculates the *lub* of all time-span (sub)trees of a designated size and makes a decision on similarity based on the similarity parameters. If fragments P and Q are similar and Q and R are too, then Summarizer infers P, Q, and R are all similar and makes them a cluster (e.g. named part A and colored red). In Figure 14, the width of each rectangular slice means the time span of a corresponding time-span tree and ranges from 12.8 (= 16 * 0.8) to 19.2 (= 16 * 1.2) beats. On the other hand, the height of each rectangular slice is one-third the height of the entire piano roll score, just because there are three kinds of clusters (parts) found. The height does not make musical sense. The intervals that are not covered by any rectangular slice mean that there is no similar fragment within the same piece. Below the GUI window, for reference, the standard notation of the result of clustering is shown (A B A C …).

4.3 Interaction with a User

The interaction with the user on the Summarizer's GUI begins with the situation shown in Figure 14 and proceeds as follows (Figure 15).

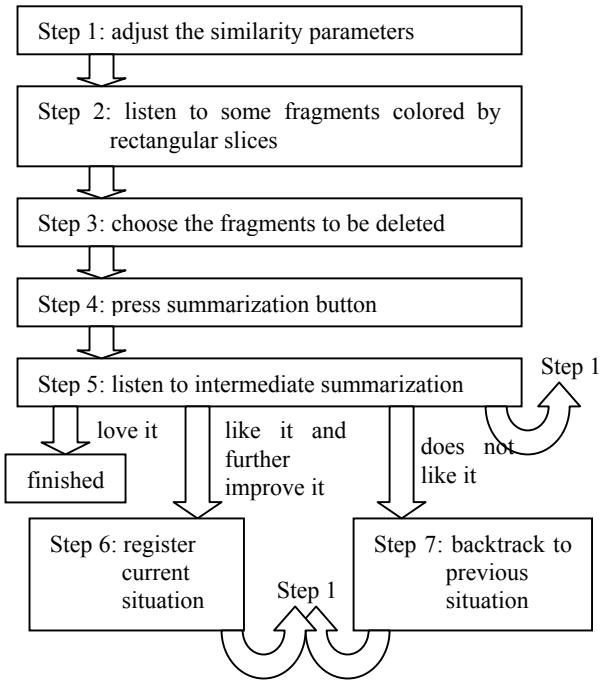

Figure 15. User's operations on Summarizer's GUI.

Upon pressing the summarization button at Step 4, the fragments chosen by the user at Step 3 are deleted, the remaining parts are concatenated, and the summarization ratio is updated. At that time, vertical black bands emerge to indicate the deleted parts (Figure 16). The 2nd and 3rd of part A, the 2nd of part B, and the 2nd and 3rd of part C are deleted, and the summarization ratio decreases to 68.75%.

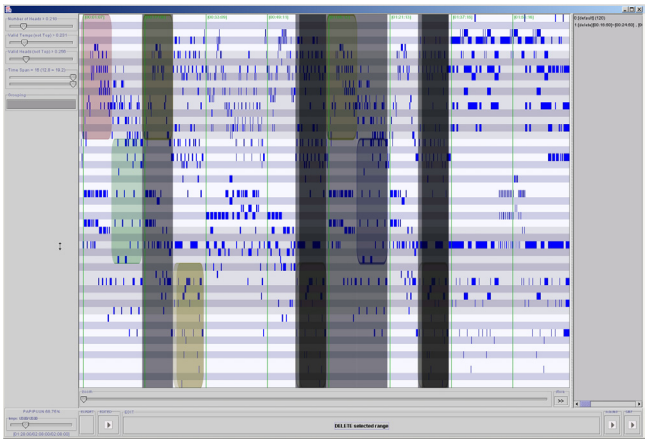

Figure 16. Snapshot with some parts deleted.

At Step 5, the user plays back an intermediate summarization and makes fine adjustments by changing the playback rate slider varied from 0.5~2.0 (1.0 means normal rate). At Step 6, a current situation consisting of the intermediate result and the values of the similarity parameters is saved for backtracking. Then, a line of the current situation is appended to the end of the operation history. The user can save a current summarization as a SMF file any time in a session, as well as upon finishing.

For instance, the user can further proceed with summarization and may change the fragment size from 16 to 4 to 2 and reach the situation in Figure 17 with the summarization ratio being 41.31%. Note that the number of lines in the operation history has increased.

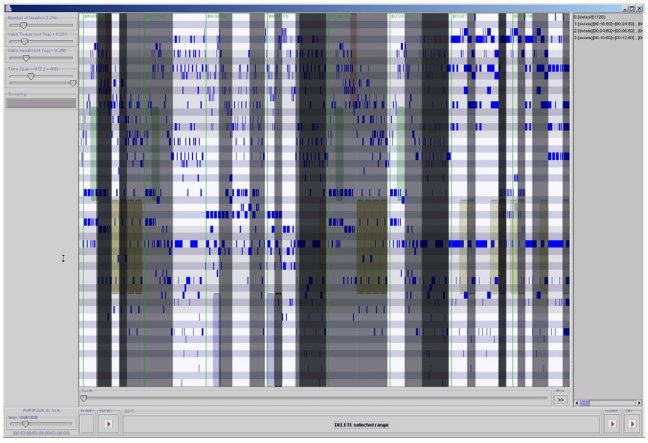

Figure 17. Snapshot of a further summarization.

5. CONCLUSION

We have not done an audition experiment with subjects, but several people who tried Papipuun reported mostly favorable impressions. However, since a summarization result highly depends on the skill with which Papipuun is manipulated and the

user's musical sense in changing the similarity parameters and choosing fragments to be skipped, it will be difficult to construct a proper experimental framework for Papipuun and evaluate it appropriately.

As mentioned in Section 1, the key issues in music summarization are (a) music representation of polyphony, (b) discovery of a piece's structure by similarity checking, (c) identification of fragments to be skipped, and (d) concatenation of selected fragments. Regarding (a), amalgamating other music theories into our framework for music representation may improve preciseness and efficiency. The module for (b) is currently implemented for manual generation. Automatizing this module is future work. Regarding (c), it may be convenient for a user to indicate parts that are not only skipped but also concatenated, although the current GUI provides only a capability for the latter. Automatizing this task is also future work. For (d), since Summarizer just concatenates fragments not to be skipped, it occasionally generates unnatural concatenations. Hence, an intelligent concatenation technique is required.

TS-Editor greatly reduces effort, but further improvements are needed. We think one is some means of supporting mechanical, routine manipulations of a time-span tree in TS-Editor. A similar idea would be useful for Summarizer's GUI as well.

Lastly, some other music summarization algorithms can be considered. One decreases the number of notes by performing the time-span reduction and fast-forwards the reduced one. We will try to integrate such algorithms into Papipuun in the future.

6. ACKNOWLEDGMENTS

We thank Mr. Takaki Odachi of Digital Art Creation for data entry. We also thank Prof. Tatsuya Aoyagi of Tsuda College for discussions in the early stage of this research.

[1] 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.

[2] In Figure 13, red lines are the branches of the time-span tree, while green lines are the temporal structures, although readers will not be able to distinguish them on the hard-copy proceedings, unfortunately.

7. REFERENCES

[1] Carpenter, B. The Logic of Typed Feature Structures. Cambridge University Press. 1992.

[2] Hirata, K., and Aoyagi, T. Musically Intelligent Agent for Composition and Interactive Performance. In Proceedings of ICMC 1999. pp.167-170.

[3] Hirata, K., and Aoyagi, T. Representation Method and Primitive Operations for a Polyphony based on Music Theory GTTM. IPSJ Journal 43, 2. pp.277-286. 2002. In Japanese.

[4] Hirata, K., and Hiraga, R. Next Generation Performance Rendering – Exploiting Controllability. In Proceedings ICMC 2000. pp.360-363.

[5] Huron, D. Perceptual and Cognitive Applications in Music Information Retrieval. In Proceedings of ISMIR 2000.

[6] Kifer, M., Lausen, G., and Wu, J. Logical Foundations of Object-Oriented and Frame-Based Languages. Journal of ACM 42, 3. 1995.

[7] Lerdahl, F., and Jackendoff, R. Generative Theory of Tonal Music. The MIT Press. 1983.

[8] Logan, B., and Chu, S. Music Summarization using Key Phrases. In Proceedings of ICASSP 2000.

[9] Plaza, E. Cases as terms: A feature term approach to the structured representation of cases. Lecture Notes in Artificial Intelligence Vol.1000. pp.265-276. Springer-Verlag. 1995.

[10] Selfridge-Field, E. Conceptual and Representational Issues in Melodic Comparison. Computing in Musicology 11, pp.3-64. 1998.

[11] Yokota, K. Towards an Integrated Knowledge-Base Management System: Overview of R&D on Databases and Knowledge Bases in the FGCS Project. In Proceedings of International Conference on Fifth Generation Computer Systems 1992. Institute for New Generation Computer Technology. pp.89-112.