JOURNAL OF NEW MUSIC RESEARCH

Volume 35, NO. 4, Pages 249-277, 2006

Implementing "A Generative Theory of Tonal Music"

By

Masatoshi Hamanaka, Keiji Hirata and Satoshi Tojo

Taylor & Francis
Taylor & Francis Group

4 Park Square, Milton Park, Abingdon
Oxfordshire, OX1 4RN, UK

Journal of New Music Research
2006, Vol. 35, No. 4, pp. 249–277

Implementing "A Generative Theory of Tonal Music"†

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

[1]University of Tsukuba, Japan; [2]NTT Communication Science Laboratories, Japan; [3]Japan Advanced Institute of Science and Technology, Japan

Abstract

This paper proposes a music analysing system called the automatic time-span tree analyser (ATTA). ATTA derives a time-span tree that assigns a hierarchy of "structural importance" to the notes of a piece of music based on the generative theory of tonal music (GTTM). Although the time-span tree has been applied in music summarization and collaborative music creation systems, these systems use time-span trees manually analysed by experts in musicology. Current systems based on GTTM cannot acquire a time-span tree without manual application of most of the rules, since GTTM does not resolve much of the ambiguity involved in the application of the rules. To solve this problem, we propose a novel computational model of GTTM that re-formalizes the rules through a computer implementation. The main advantage of our approach is that we can introduce adjustable parameters, which enables us to assign priorities to the rules. Our analyser automatically acquires time-span trees by configuring the parameters that cover 17 out of 26 GTTM rules for constructing a time-span tree. Experimental results show that after these parameters were tuned, our method could outperform a baseline performance. We hope to distribute the time-span tree analyser as a tool for various musical tasks, such as searching and arranging music.

1. Introduction

Musical activities reflect many aspects of human intelligence. These activities encompass not only composition, arrangement, and performance but also listening, dancing to music, and even selecting CDs at Amazon or compiling playlists for your friends. The relevant research fields include musicology, physiology, psychology, artificial intelligence, and sociology.

Music theory, in particular, focuses on a piece written on a score, gives us the methodology for analysing and understanding a piece, and explains deep structure, musical knowledge, experiences, and skills in a comprehensive way. We believe that implementing music theory on a computer would bring great benefits because it could provide a theoretical basis for developing a support system for the above musical activities. For instance, such advantages include performance rendering (Todd, 1985; Widmer, 1993; Hirata & Hiraga, 2003) and music summarization for practical use when displaying the results of a music information retrieval system (Hirata & Matsuda, 2003).

To get a music theory to operate on a computer, however, we must overcome some widely recognized fundamental difficulties. One is giving an ambiguous concept a firm definition, and the other is supplementing the lack of necessary concepts (externalization). For example, we may easily decide whether two melodies are similar to each other, but in general each of us likely makes a different decision, and it is difficult to fully explain why the melodies are similar. To write a program that replicates a human checking melodic similarity, we have to cope with the problem commonly known as knowledge acquisition (Russell & Novig, 2002). Marsden (2005) states this problem from a musicology point of view as meeting the following requirements of a representation system for structural music manipulation:

- Musical symbols, objects, and concepts are all well defined; a new one is defined by predefined ones.

†An earlier version of this paper won the Journal of New Music Distinguished Paper Award at the International Computer Music Conference, 2005.

Correspondence: Masatoshi Hamanaka, Department of Intelligent Interaction Technologies University of Tsukuba, 1-1-1 Tennoudai, Tsukuba, Ibaraki 305-8573, Japan. E-mail: hamanaka@iit.tsukuba.ac.jp

DOI: 10.1080/09298210701563238 © 2006 Taylor & Francis

- Musical symbols, objects, and concepts are all grounded to relevant ones in the real world; there must be a consistent, well-segmented correspondence between musical symbols, objects, and concepts in a representation system and those in the real world.

We think these requirements can play an important role in overcoming the above difficulties and mechanizing music theory.

Although many music theories have been proposed (Cooper & Meyer, 1960; Schenker, 1979; Lerdahl & Jackendoff, 1983; Narmour, 1990; Cope, 1995; Temperley, 2001), each theory is differently motivated, and the insight used as a starting point is different, as are the fundamental concepts and models. Furthermore, each theory is presented in a different description style; for example, one is presented totally in a descriptive style using sample scores, and another in a procedural style showing program fragments. Therefore, to offer efficient and precise definitions of the musical symbols, objects, and concepts involved in defining similarity while still satisfying the above requirements, we have to carefully examine music theories for their potential to operate on a computer and then select one as a starting point.

We believe that the Generative Theory of Tonal Music (GTTM) (Lerdahl & Jackendoff, 1983) is the most promising theory for mechanizing and developing a support system, thanks to three of its features. The first is the use of rules to describe the musical insight and knowledge obtained by investigating in detail the musical structures and relations occurring in a piece of music. The rules of GTTM accumulate fundamental musical phenomena and concepts, define the relations between them, and prescribe the conditions to be satisfied as well as the desired situation. Originally, Lerdahl and Jackendoff employed the form of rules to express their results in a manner that ensures human readability, since rules can properly segment and organize musical introspection and knowledge so that a human can easily understand them. In addition, such rules can be translated into a computer program due to the modularity of knowledge.

The second feature is that GTTM is designed to consistently represent the multiple aspects of music in a single framework. This feature is important in the sense that when GTTM is executed in a mechanical way, contradiction should be prevented as much as possible. For instance, if we imagine a simple operation that splits a melody, the condition of applying the split operation may vary depending on the relevant musical structure. Therefore, it is preferred that the splitting position of a melody and that of an ornamented melody be essentially identical. Unless a system of consistent operations in terms of melody, rhythm, and harmony is developed, the resulting splitting positions may be different.

The third feature is that GTTM has been developed based on the concept of reduction. We believe that the proper way of taking into account deep structures is to adopt the concept of reduction (Selfridge-Field, 1998; Hirata & Aoyagi, 2003). Reduction, in general, connects an original element and its ornamented one (complicated one), and it corresponds to an "is-a" relation, the most fundamental relation in knowledge representation. Accordingly, deep structure would emerge from the network of concepts connected by relations involving "is-a," where many knowledge representation techniques are available (Russell & Novig, 2002). In contrast, conventional approaches to representing and comparing music mostly focus on surface information and do not introduce the "is-a" relation. Thus, they unfortunately cannot handle deep structures or high-level musical knowledge properly.

Furthermore, Lerdahl and Jackendoff also recognized the advantages of mechanizing GTTM to some extent:

> Our theory cannot provide a computable procedure for determining musical analyses. However, achieving computability in any meaningful way requires a much better understanding of many difficult musical and psychological issues than exists at present. (Lerdahl & Jackendoff, 1983, p. 55)

We think that "many difficult musical and psychological issues" would encompass what we are trying to solve in this paper.

In this work, we extend GTTM by full externalization and parameterization (Section 3.3.1) and propose exGTTM. This externalization in mechanizing GTTM includes introducing an algorithm for generating a hierarchical structure of the time-span tree in the mixed manner of top-down and bottom-up. Such an algorithm has not been presented for GTTM, and in fact it seems that researchers have not even recognized the need for this type of algorithm. The parameterization includes introducing a parameter controlling the priorities of rules to avoid a conflict among the rules as well as the parameters for controlling the shape of the hierarchical time-span tree. It has been suggested that such control parameters are required in GTTM, but they have not been explicitly presented. On the other hand, we restrict GTTM and implement a subset of GTTM because mechanization has priority over all others. For example with regard to the restrictions, only a monophony is handled, and harmony is not taken into account. We totally sum up the detail on restrictions in Section 3.3.

Here, note that we distinguish two kinds of ambiguity in music analysis: one involves music understanding by humans, and the other concerns the representation of a music theory. The former kind of ambiguity derives from the ambiguity of music itself. The following is quoted from the GTTM book:

> In music, ... grammaticality per se plays a far less important role, since almost any passage of music is potentially vastly ambiguous – it is much easier to construe music in a

multiplicity of ways. The reason for this is that music is not tied down to specific meanings and functions, as language is. In a sense, music is pure structure, to be "played with" within certain bounds. The interesting musical issues usually concern what is the most coherent or "preferred" way to hear a passage. (Lerdahl & Jackendoff, 1983, p. 9)

For the latter type of ambiguity, related to GTTM, no concept needed for mechanization has been presented, or it has been presented only in an implicit way. Therefore, due to the former kind of ambiguity, we assume there are more than one correct result. We avoid the latter kind of ambiguity as much as possible by full externalization and parameterization.

The significance of full externalization and parameterization is twofold: precise controllability and coverage of the human results. Whenever we find a correct result that exGTTM cannot generate, we introduce new parameters to exGTTM and give them proper values so that it can generate the correct result. In this way, we repeatedly externalize and introduce new parameters until we can obtain all of the results that humans consider correct. In total, we have introduced 15 parameters for grouping-structure analysis, 18 for metrical-structure analysis, and 13 for time-span reduction. It is eventually guaranteed that for every piece in a given set of pieces, by assigning certain values to these parameters, exGTTM can generate a correct analysis result. In other words, exGTTM can precisely control the analysis result by adjusting the parameter values, thus covering all of the correct analysis results. We think that toward a fully automatic GTTM analyser, full externalization and parameterization is an important intermediate step. Moreover, we think that full externalization and parameterization are significant in light of the testability of music theories. Temperley pointed this out as follows:

If the parameters of the rules can be specified, the output of the rule system for a given input can be determined in an objective way, making the theory truly testable. (Temperley, 2001, p. 14)

The organization of the paper is as follows. In Section 2, we mention related works, and in Section 3, we briefly describe music theory GTTM, discuss the problems and how to solve them when implementing GTTM, and propose exGTTM as an extension of GTTM. In Sections 4, 5 and 6, we present the detailed algorithms for implementing the grouping-structure analyser, metrical-structure analyser, and time-span tree analyser. In Section 7, we explain the implemented system called ATTA and its features, and in Section 8, we evaluate the performance of the implemented system. Finally in Section 9, we conclude with a summary and overview of future work.

2. Previous work

Our primary concern is implementing a music theory. We do not compare GTTM with other music theories on their own merits, since musicological comparison is out of the scope of this paper. The preference rule systems (PRS) (Temperley, 2001) approach, along with its practical implementation, the Melisma system of Sleator and Temperley, is a pioneering work in implementing music theory. PRS analyses music from six aspects: meter, melodic phrase, counterpoint, pitch spelling, harmony, and key. For each analysis, Temperley presents a set of well-formedness rules and the preference rules. The former defines the possible structures considered legal, while the latter chooses the optimal analysis out of the possible ones, as GTTM described in Section 3.1. In the following, we briefly describe the implementation strategy of PRS and the Melisma system.

The Melisma system basically employs dynamic programming to find the best-so-far partial analyses at every intermediate stage, and it finally reaches an entire analysis with the highest score. Temperley addresses that there are two kinds of problems in implementing the entire analysis. The first problem is how the PRS program is to evaluate individual possible analyses. The program analyses a piece by assigning a numerical score to each analysis, which reflects how well it satisfies the preference rules. The second problem is how the program evaluates all possible well-formed analyses of a piece according to the preference rules and then finds the most plausible analysis, possibly with the highest score.

Let us examine an example of handling the first problem in meter analysis. To calculate the strength of a unit segment, we assume each note contained in the segment contributes a value, called effective length, which is either the maximum of its duration and its registral IOI (inter-onset interval to the following note within the line of texture) or 1.0, whichever is smaller. The total strength of a segment is calculated by the square root of its number of notes times its average effective length. Thus, the contribution of each note to the total strength is predetermined and fixed.

For the next example, we handle the first problem in melodic phrase analysis. The first step in devising the implementation is to determine a way of evaluating analyses by three rules: gap, phrase length, and parallelism. Here, correct weights for the three rules relative to one another were determined by a trial-and-error process beforehand, and for a given phrase, the score of each sub-phrase is further weighted by the length of the sub-phrase.

When the program searches for a globally optimal analysis based on the results of the local analyses, it must consider global analyses rather than simply choosing the best analysis for each segment in isolation.

Consequently, the second problem arises. Usually, even for a monophonic short melody, the number of possible local analyses may grow exponentially with the number of segments, and the size of the best-so-far analysis becomes extremely large. The Melisma system suppresses the explosion of analyses by properly pruning less significant analyses by dynamic programming. Temperley argues that, to some extent, this searching process reflects the human moment-to-moment cognitive process of revision, ambiguity and expectation when listening to music.

As described above, in PRS, the weights of rules and parameter values are fixed during an entire analysis. Thus, a conflict of rule applications never occurs, and some analysis can be obtained in any event. However, the conflict is not always resolved appropriately, and a true analysis is not always acquired. We have an intuition that the proper rule weights and parameter values must vary dynamically, depending on the part of the piece being analysed.

In our research, we do not consider the cognitive process itself. Since the GTTM rules already represent the cognitive process at a conceptual level, we focus on implementing the GTTM rules.

In Melisma, a hierarchical metrical structure is acquired as follows. First, the best location for a tactus beat within a range of 400 to 1600 ms is identified, using a variant of the dynamic programming technique. This forms the preferred tactus level for the piece that emerges from the meter analysis rules. Then, the program derives the upper levels then the lower levels; that is, the hierarchy is acquired in a middle-out manner. The regularity rule is given priority, at the upper levels, i.e. imposing a penalty for changes in the number of beats, rather than at the tactus level. Note that at the top-most level (the whole note level), to avoid the side effect of the length rule (preferring long notes as strong beats), Melisma just ignores the length rule as heuristics. Temperley, however, argues that a substantial solution to the side effect is introducing a grouping rule.

In our research, to acquire the hierarchies of the GTTM analyses, we adopt a strategy of taking into account global and local information at the same time, in contrast to the middle-out approach in the meter analysis of PRS. To eventually obtain an optimal solution that satisfies as many preference rules as possible, we think our strategy is advantageous.

Stammen and Pennycook (1994) implemented only the basic rules for a grouping structure, and a hierarchical grouping structure cannot be obtained. Nord (1992) implemented grouping, metrical, and time-span analyses, but the rule applications are done by hand.

Among the previous works not directly related to GTTM, the techniques for melody segmentation are well developed, and there have been many attempts to propose algorithms for melody segmentation at boundaries (Stammen & Pennycook, 1994; Cambouropoulos, 2001; Temperley, 2001; Ferrand et al., 2003). Conventional methods focus on identification of local boundaries but do not pay much attention to obtaining a hierarchical grouping structure for acquiring a time-span tree like GTTM.

In research using a Voronoi diagram drawn on a piano roll score (Hamanaka & Hirata, 2002), a polyphony is analysed in a hierarchical way. However, since the analysis lacks a musical basis, there are cases in which no appropriate hierarchical grouping structure is obtained. The methods based on beat tracking (Rosenthal, 1992; Goto, 2001) can also only acquire a hierarchical metrical structure in one measure, since they cannot consider such larger metrical structures as two measures, four measures, and so on.

3. Implementing GTTM on a computer

Among music theories, GTTM is tractable for implementation on a computer, but many tasks remain toward building a working system. Toward this end, we briefly summarize the organization of GTTM and present the implementation problems involved. Then, we propose strategies for resolving these problems.

3.1 Generative theory of tonal music

GTTM is a music theory for describing the insight of an "experienced listener", and it consists of four subtheories: grouping-structure analysis, metrical-structure analysis, time-span reduction, and prolongational reduction.

The grouping-structure analysis hierarchically divides a series of notes of a homophony into phrases or motives. It seems that a singer looks for breathing points when singing a long melody. These groups are graphically presented as several levels of arcs below a music staff (figure 1). The metrical-structure analysis identifies strong and weak beats at each metrical level: quarter note, half note, whole note, two measures, and four measures. Essentially, it looks for timings at which a

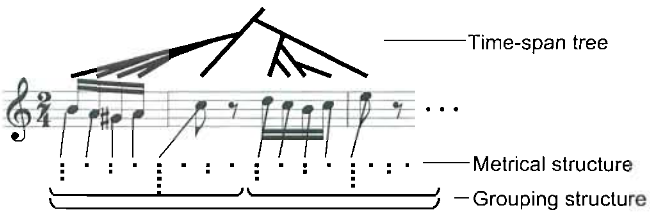


Fig. 1. Grouping structure, metrical structure, and time-span tree.

listener beats time with his/her hands to music or a conductor swings his/her baton. Strong beats are illustrated by dots in multiple levels below the music staff (figure 1). Time-span reduction distinguishes important parts of a melody from unimportant ones and yields a binary tree, called a time-span tree, so that each structurally important note belongs to a stem at every level. Figure 2(a) shows a melody and its corresponding time-span tree, where a single note, called a head (note C4 shown in figure 2(b)), represents the time span denoted by <---> containing the two notes. The prolongational reduction generates a tree structure representing subordinate relationships between chords by explicitly indicating harmonic retention and change.

Each sub-theory of GTTM is described by two kinds of rules: a well-formedness rule prescribes the conditions and constraints that must always be satisfied; a preference rule prescribes which structure is preferable among the structures satisfying the well-formedness rules. Thus, depending on the situation, some preference rules hold and others do not.

Among the four sub-theories, this paper presents the methodology for implementing the grouping-structure analysis, metrical-structure analysis, and time-span reduction (Hamanaka et al., 2004, 2005a,b). Since the prolongational reduction is still evolving and controversial, we do not implement it at present.

3.2 Problems in implementing GTTM

As described above, when we make GTTM rules operate on a computer, we have to make an ambiguous concept a firm definition and supplement the lack of concepts. Roughly speaking, these tasks correspond to parameterization and externalization, respectively. Here, we discuss the problems occurring in the process of parameterization and externalization.

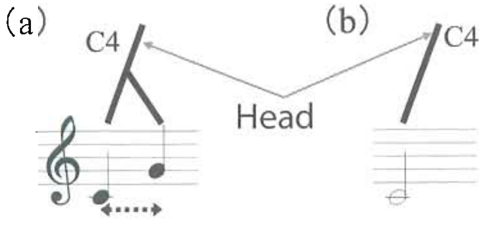


Fig. 2. Simple example of time-span tree.

3.2.1 Ambiguous rule definition

As an example, the following is Grouping Preference Rule 4 (GPR4), regarding the extent to which GTTM intensifies the effects of GPR2 and GPR3:

> Where the effects extracted by GPR2 and GPR3 are relatively more pronounced, a larger-level group boundary may be placed

Here, GPR2 prescribes the relationship of onset (attack) and offset (release) timings and a group boundary. GPR3 prescribes the relationship of pitch interval, dynamics, note duration, and group boundary. That is, there are many factors influencing the determination of a group boundary. Accordingly, the meaning of the phrase "relatively more pronounced" in GPR4 is ambiguous, because we do not know how to normalize the effects of the factors or how to compare the effects with each other. Moreover, the phrase "may be placed" implies that at one time GPR4 holds, while at another time it does not. However, we do not know in what particular situation GPR4 holds.

3.2.2 Conflict among preference rules

Since preference rules stipulate situations where they preferentially hold, they may often conflict. However, there are no rules for resolving preference rule conflicts in GTTM.

For example, Figure 3 shows the situation of conflicting preference rules. Here, GPR3a identifies a leap point in the melody as a group boundary (notes 3–4, notes 7–8, and notes 11–12), while GRP6 identifies an iterating point as a group boundary (notes 4–5, notes 8–9, and notes 11–12). Thus, GPR1 is as follows:

> Avoid analyses with very small groups

That is, there is a constraint to prevent a group consisting of a single note. Therefore, GPR3a and GPR6 conflict with each other at notes 4 and 8. If both group boundaries were adopted, we would obtain single-note groups and GPR1 would be violated. Consequently, we should adopt either GPR3a or GPR6 at notes 4 and 8; however, we do not know which GPR should be given a high priority.

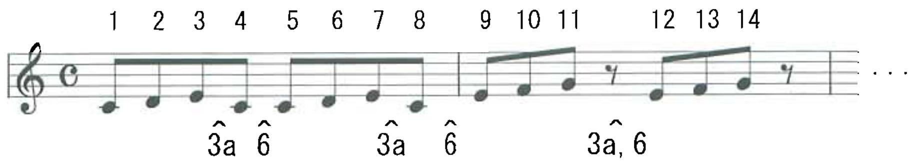


Fig. 3. Simple example of conflict between rules.

3.2.3 Lack of working algorithm

Knowledge represented in the rule form is in general considered declarative, which is advantageous in the sense that a knowledge programmer does not need to take into account an algorithm for reasoning. Instead, a system is required to perform automatic reasoning on the knowledge declaratively described. In GTTM, unfortunately, there are few descriptions of the reasoning and working algorithms needed to compute analysis results.

For example, GPR6 is as follows:

> Where two or more segments of the music can be construed as parallel, they preferably form parallel parts of groups.

GPR6 is motivated by the fact that the beginning and the end of a parallel segment may be heard as a group boundary. Therefore, we should design an algorithm to indicate the beginning and the end of every parallel segment occurring in a piece. Furthermore, the depth of a boundary must reflect the properties of a parallel segment, where the properties include the length, employing either pitch-oriented matching or timing-oriented matching, and the weighting of either the beginning or the end. As a result, we will possibly obtain the bottom graph of Figure 10 in Section 4.1.4. However, the GPR6 statement does not provide any procedural information for generating such a function.

Procedural information for constructing hierarchical structures for grouping-structure analysis and time-span reduction is not sufficient in GTTM. For example, GPR4 mentions which placement of a larger-level group boundary is preferable but does not state an algorithm for this placement. Moreover, when constructing a grouping structure, a working algorithm has to take into account not only GRP2 and GPR3 but also GPR5 and GPR6. GRP2 and GPR3 concern local structures and generate a grouping structure in the bottom-up manner; in contrast, GPR5 and GPR6 concern global structures and generate a grouping structure in the top-down manner. It would be difficult to design an effective algorithm incorporating these GPRs consistently. As another example, the preference rules of time-span reduction (TSRPRs) only mention which head should be selected, not how. To describe how to select a head, a working algorithm has to appropriately identify the set of head candidates from which a head is selected and then integrate bottom-up and top-down head selections, similarly with GPRs.

3.3 Solution: proposal of exGTTM

We present the design strategy of a machine-executable extension of GTTM, exGTTM. As described, the fundamental difficulties involve giving an ambiguous concept a firm definition and supplementing the lack of concepts. To overcome these difficulties, we employ full-externalization and parameterization to yield precise controllability and coverage of the human results.

We are implementing a subset of GTTM. Before moving into details, it would be helpful to the readers to sum up the restrictions in implementing GTTM.

- Only a monophony is handled.
- Harmony (key and chord progression) is not taken into account.
- Only ordinary heads occur in a time-span tree.
- There are no feedbacks from the time-span reduction to the grouping- and metrical-structure analyses.
- The prolongational reduction is not mechanized.
- The input representation is a list of notes that corresponds to a monophonic piano roll.
- We divide the entire problem into identifying the domain of all possible answers and searching for the most preferred one; here, we only deal with the former sub-problem.
- We focus on implementing GTTM, but it does not consider the human's process of perceiving and recognizing music.

Paradoxically speaking, what we have not implemented enables us to implement GTTM.

3.3.1 Full externalization and parameterization

We appropriately supply lacking parameters and make implicit parameters explicit[1]. The parameters introduced by exGTTM are categorized into three categories: identified, implied, and unaware.

For the first category, a parameter is identified in GTTM but is not assigned concrete values. Hence, we valuate such a parameter. For example, since the resulting value of the GPR2a application, D_{GPR2a}, is binary, if it holds, D_{GPR2a} makes 1, otherwise 0. On the other hand, since GPR6 is held indefinitely, the resulting value of GPR6, D_{GPR6}, also varies continuously between 0 and 1.

For the second category, a parameter is implied in GTTM. Hence, we make it explicit. For example, to resolve the preference rule conflict, we introduce parameters to express the priority for each preference rule (S^{GPR_R} in Table 1, S^{MPR_R} in Table 2, and S^{TSRPR_R} in Table 3). Since each preference rule has its own priority, all of the priority patterns are realized. This is an example of full-parameterization.

For the third category, we need to complement parameters that are not recognized in the original theory,

[1]In the paper, the word "parameter" is used not only for parameters used in controlling a system externally but also for internal variables (intermediated variables) connecting submodules.

Table 1. Fifteen adjustable parameters for grouping-structure analyser.

Parameters	Description
S_{GPR_R} $(0 \leq S_{\mathrm{GPR}_R} \leq 1)$	Strength of each rule. The larger the value is, the stronger the rule acts in (19) and (23). $R \in \{2a, 2b, 3a, 3b, 3c, 3d, 4, 5,$ *and* $6\}$.
$\hat{\sigma}$ $(0 \leq \hat{\sigma} \leq 0.1)$	Standard deviation of a Gaussian distribution, the average of which is the boundary by GPR5. The larger the value is, the wider its skirt becomes in (14).
W_m $(0 \leq W_m \leq 1)$	Balance between temporal similarity of attack points and that of pitch difference in GPR6. The larger the value is, the more the system estimates the pitch difference in (15).
W_l $(0 \leq W_l \leq 1)$	Weight for the length of parallel phrases. The larger the value is, the more the length of parallel phrases is prioritized in GPR6 in (15).
W_s $(0 \leq W_s \leq 1)$	Balance determining whether the note i becomes the ending note of a group or the beginning note of the following group in GPR6. The larger the value is, the more the note tends to be the ending note in (16).
T_{GPR4} $(0 \leq T_{\mathrm{GPR4}} \leq 1)$	Threshold at which the effects of GPR2,3 are considered to be salient in GPR4. The smaller the value is, the more probably GPR4 is applied in (13).
T^{low} $(0 \leq T^{\mathrm{low}} \leq 1)$	Threshold in the lower-level boundary. The smaller the value is, the more salient the boundary becomes.

Table 2. Eighteen adjustable parameters for metrical-structure analyser.

Parameters	Description
S_{MPR_R}	Strength of each rule. The larger the value is, the stronger the rule acts in (34), (35) and (38). $R \in \{1, 2, 3, 4, 5a, 5b, 5c, 5d, 5e,$ *and* $10\}$
$\acute{W}_m$ $(0 \leq \acute{W}_m \leq 1)$	Balance between temporal similarity of attack points and that of pitch difference in MPR1. The larger the value is, the more the system estimates the pitch difference.
$\acute{W}_l$ $(0 \leq \acute{W}_l \leq 1)$	Weight for the length of parallel phrases. The larger the value is, the more the length of parallel phrases is prioritized in MPR1.
$\acute{W}_s$ $(0 \leq \acute{W}_s \leq 1)$	Balance determining whether the note i becomes the ending note of a group or the beginning note of the following group in MPR1. The larger the value is, the more the note tends to be the ending note.
T_{MPR_R}	Value of the threshold that decides whether each rule is applicable in (25), (28), (29), (30), (31). $R \in \{1, 4, 5a, 5b,$ *and* $5c\}$

Table 3. Ten adjustable parameters for time-span tree analyser.

Parameters	Description
S_{TSRPR_R}	Strength of each rule. The larger the value is, the stronger the rule acts in (46) and (47). $R \in \{1, 2, 3, 4, 5a, 5b, 5c, 5d, 5e,$ *and* $10\}$
$\grave{W}_m$ $(0 \leq \grave{W}_m \leq 1)$	The balance between the temporal similarity of attack points and that of the pitch difference in TSRPR4. The larger the value is, the more the system estimates the pitch difference.
$\grave{W}_l$ $(0 \leq \grave{W}_l \leq 1)$	The weight for the length of parallel phrases. The larger the value is, the more the length of parallel phrases is estimated in TSRPR4.
$\grave{W}_s$ $(0 \leq \grave{W}_s \leq 1)$	The balance determines whether the note i becomes the ending note of a group or the beginning note of the following group in TSRPR4. The larger the value is, the more the note tends to be the ending note.

since some of them may nearly lack any musicological meanings. For example, GPR6 in exGTTM needs to add parameters for controlling the properties of parallel segments, including the weights for pitch-oriented matching or timing-oriented matching.

We add a comment to the domain of intermediate variables, denoted as D and B. The domain of all the intermediate variables is constrained within the range of 0 to 1, and for this purpose, those variables are normalized at every computing stage. Thanks to this property, exGTTM can flexibly combine any intermediate variables (and possibly parameters) and cascade as many weighted-mean calculations as needed. Accordingly, this facilitates precise controllability.

3.3.2 Algorithm for acquiring hierarchy

Among issues that require working algorithms, the problems for acquiring hierarchical structures in the grouping- and metrical-structure analyses and the time-span tree reduction can be all regarded as a constraint-satisfaction problem (CSP). This is because only the properties to be satisfied for the hierarchical structures are represented in the form of a rule, that is, no constraint nor order of generating hierarchical structures is determined in advance.

The constraints stipulated by the GTTM rules are divided into two categories: local and global. The former include GPR2 (proximity) and TSRPR1 (strong metrical position), and the latter GPR5 (symmetry) and MPR1 (parallelism). We need to handle global constraints carefully when generating hierarchical structures. For the example of GPR5 in Figure 4, given a group at Layer 1, an inner boundary likely occurs around the centre of the group, that is, either between Notes 1 and 2 or Notes 2 and 3. Here, we can consider two cases as follows. In Case 1, the boundary between Notes 1 and 2 is selected, taking into account the effects of some other rules. Then in each subgroup in Layer 2, the inner boundary of the subgroup may occur in the left-hand side of a centre note. On the other hand, in Case 2, the boundary between Notes 2 and 3 is selected. Therefore, the inner boundary may occur in the right-hand side of a centre note. Consequently, in computing GPR5, the boundary position determined influences the identifications of remote boundaries in lower layers, and we have to take into account up-to-date global information every time. That is, a global constraint is inevitably dynamic.

Based on the above consideration, we are developing algorithms for generating hierarchical structures for exGTTM so that nodes are generated either from the bottom-most nodes or the top-most node incrementally and that every time the nodes at a layer are calculated,

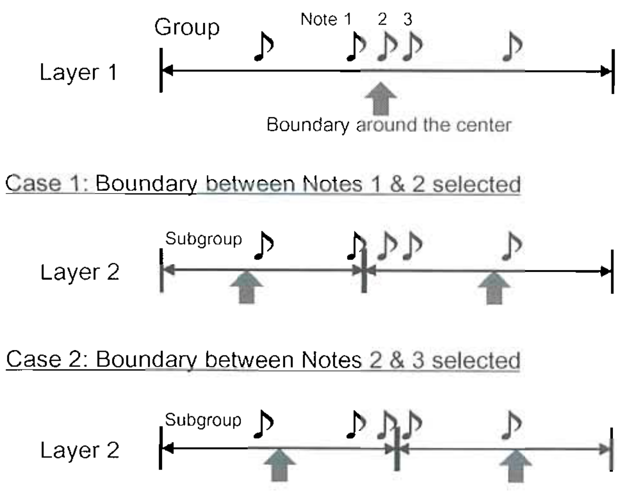

Group
Note 1 2 3
Layer 1
Boundary around the center
Case 1: Boundary between Notes 1 & 2 selected
Subgroup
Layer 2
Case 2: Boundary between Notes 2 & 3 selected
Subgroup
Layer 2

Fig. 4. Global constraints by GPR5.

global information is re-calculated before moving onto an adjacent layer. We will describe these algorithms in more detail later (Sections 4.2, 5.2 and 6.2).

4. Grouping analysis

The algorithm of grouping analysis consists of the following steps.

(1) Regard the whole piece as a group.
(2) Apply those rules that are applicable to local structures.
(3) Calculate local strength as boundaries.
(4) Apply phrasal rules.
(5) Calculate phrasal boundaries.
(6) Divide the entire piece into two.
(7) Repeat steps 4 to 6 as long as there is a local boundary.

Using this procedure, we can regulate the order of rule application and combine phrasal and local structuring.

We show the procedure in Figure 5. We adopt MusicXML (Recordare, 2007a) for the input format since it is now prevalent for composition, analysis, and search; moreover, it can be easily converted to other formats. However, exGTTM does not use the information of key and meter provided by MusicXML. In addition, we have designed GroupingXML as an output format of the grouping structure.

In this procedure, we restrict the target of analysis to monophonic music.[2]

4.1 Application of grouping preference rules

In this section, we explain ten grouping preference rules (GPRs), i.e. GPR$_R$ ($R \in \{1, 2a, 2b, 3a, 3b, 3c, 3d, 4, 5,$ *and* $6\}$). The possibility that the ith transition becomes a boundary by the Rth rule is denoted by $D_{\mathrm{GPR}_R}(i)(0 \leq D_{\mathrm{GPR}_R}(i) \leq 1, R \in \{1, 2a, 2b, 3a, 3b, 3c, 3d, 4, 5, \textit{ and } 6\})$. In this study, we introduce the 15 adjustable parameters shown in Table 1. Figure 6 shows the relation between the grouping preference rules and the parameters. Besides the above ten rules, GTTM includes GPR7 (time-span and prolongational stability), which prefers those structures by which the time-span reduction or the prolongational reduction become more stable. However, this rule requires the information of later processes, such as time-span/prolongational reductions, to be sent back to the earlier processes, and its details are not given in GTTM (Lerdahl & Jackendoff, 1983). Consequently, we have not included GPR7 in this study.

[2]Monophonic music consists only of a melody line that is not accompanied by chords.

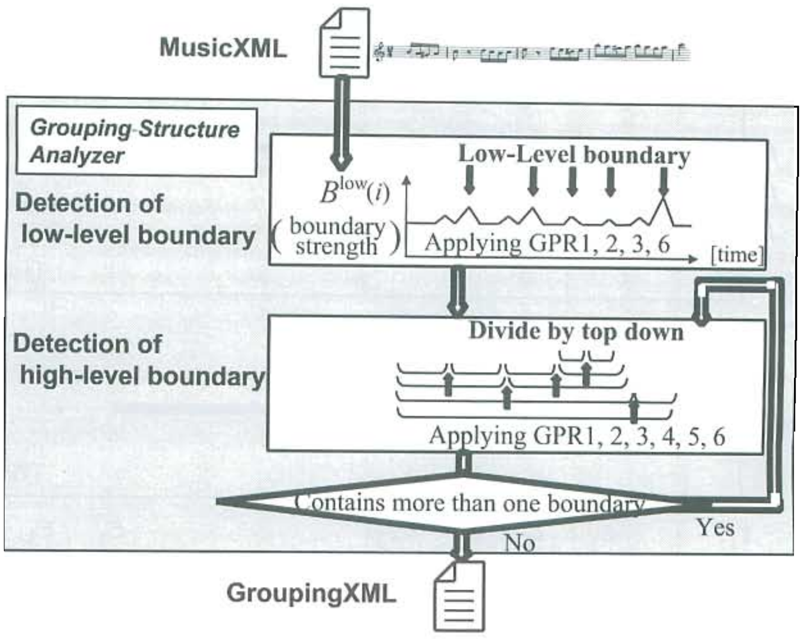

MusicXML
Grouping Structure Analyzer
Detection of low-level boundary
Low-Level boundary
$B^{low}(i)$
(boundary strength)
Applying GPR1, 2, 3, 6
[time]
Detection of high-level boundary
Divide by top down
Applying GPR1, 2, 3, 4, 5, 6
Contains more than one boundary
Yes
No
GroupingXML

Fig. 5. Processing flow of grouping structure analyser.

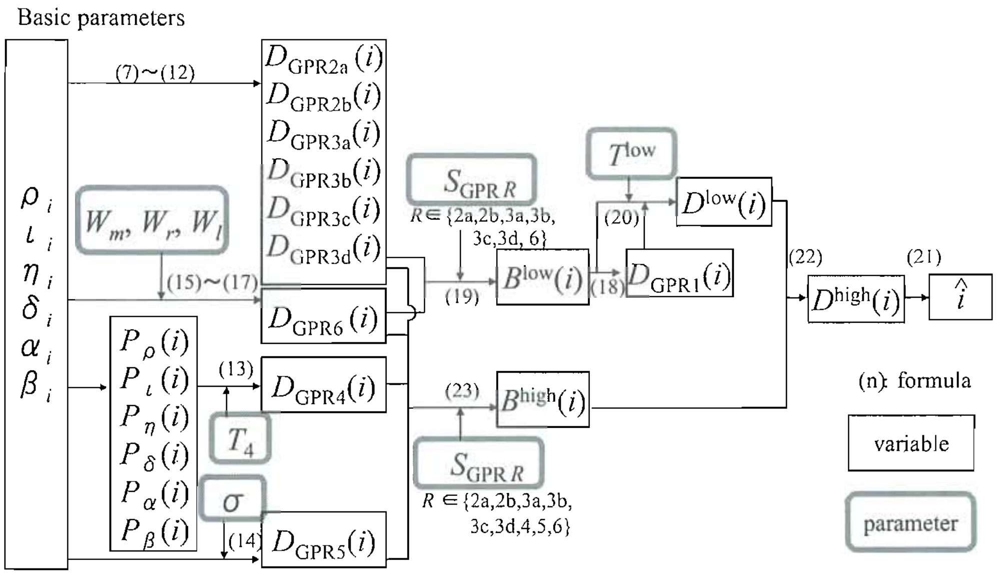

Basic parameters
ρ_i ι_i η_i δ_i α_i β_i
(7)~(12)
$D_{GPR2a}(i)$ $D_{GPR2b}(i)$ $D_{GPR3a}(i)$ $D_{GPR3b}(i)$ $D_{GPR3c}(i)$ $D_{GPR3d}(i)$
W_m, W_r, W_l
(15)~(17)
$D_{GPR6}(i)$
$P_\rho(i)$ $P_\iota(i)$ $P_\eta(i)$ $P_\delta(i)$ $P_\alpha(i)$ $P_\beta(i)$
(13)
$D_{GPR4}(i)$
T_4
σ
(14)
$D_{GPR5}(i)$
$S_{GPR\,R}$
$R \in \{2a,2b,3a,3b,3c,3d,6\}$
(19)
$B^{low}(i)$
(18)
$D_{GPR1}(i)$
T^{low}
(20)
$D^{low}(i)$
(22)
(21)
$D^{high}(i)$
$\hat{i}$
(23)
$B^{high}(i)$
$S_{GPR\,R}$
$R \in \{2a,2b,3a,3b,3c,3d,4,5,6\}$
(n): formula
variable
parameter

Fig. 6. Relationship between parameters and GPRs.

4.1.1 Calculation of basic parameters

We first calculate six basic parameters from MusicXML, that is,

ρ_i	the offset-to-onset interval (OOI)
ι_i	the inter-onset interval (IOI)
η_i	the register (pitch difference)
δ_i	the dynamics difference
α_i	the difference in the ratio between the formal length of the note on the score and its real duration in performance
β_i	the difference in duration

In Figure 7, τ_i is the formal length of the ith note on the score, ε_i is the formal offset time, $\hat{\varepsilon}_i$ is the real offset time, f_i is the pitch, and υ_i is the dynamic. The time unit is that of a quarter note, and the pitch is measured by a semitone (MIDI note number). We formally represent these basic parameters mathematically.

$$\rho_i = \begin{cases} \tau_{i+1} - \hat{\varepsilon}_i & \text{if } \tau_{i+1} - \hat{\varepsilon}_i \geq 0, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

$$\iota_i = \tau_{i+1} - \tau_i, \tag{2}$$

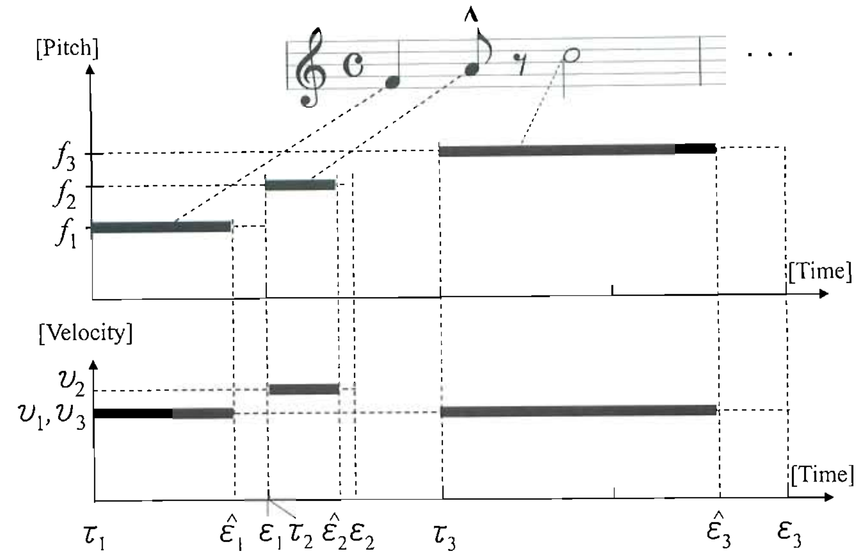

[Pitch]
f_3
f_2
f_1
[Time]
[Velocity]
υ_2
υ_1, υ_3
[Time]
τ_1
$\hat{\varepsilon}_1$ ε_1 τ_2 $\hat{\varepsilon}_2$ ε_2
τ_3
$\hat{\varepsilon}_3$ ε_3

Fig. 7. Calculation of basic parameters.

$$\eta_i = f_{i+1} - f_i, \tag{3}$$

$$\delta_i = |\upsilon_{i+1} - \upsilon_i|, \tag{4}$$

$$\alpha_i = \left| \frac{\varepsilon_{i+1} - \tau_{i+1}}{\hat{\varepsilon}_{i+1} - \tau_{i+1}} - \frac{\varepsilon_i - \tau_i}{\hat{\varepsilon}_i - \tau_i} \right|, \tag{5}$$

$$\beta_i = |\iota_{i+1} - \iota_i|. \tag{6}$$

4.1.2 Application of GPR2, 3, and 4

GPR2, 3, and 4 are applied to the four consecutive notes $n_1 C n_2 C n_3 C n_4$. The ith transition, that is, the interval between the ith note and the $i+1$th, can be either a boundary ($D_{\mathrm{GPR}_R}(i) = 1$) or not ($D_{\mathrm{GPR}_R}(i) = 0$).

GPR2a (slur/rest) recognizes a boundary if the offset-to-onset interval (OOI) of n_2 and n_3 is longer than that of n_1 and n_2 and that of n_3 and n_4. Thus, GPR2a is formalized as follows.

$$D_{\mathrm{GPR2a}}(i) = \begin{cases} 1 & \text{if } \rho_{i-1} < \rho_i \text{ and } \rho_i > \rho_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

GPR2b (attack point) recognizes a boundary if the inter-onset interval (IOI) of n_2 and n_3 is longer than that of n_1 and n_2 and that of n_3 and n_4. Thus, GPR2b is formalized as follows.

$$D_{\mathrm{GPR2b}}(i) = \begin{cases} 1 & \text{if } \iota_{i-1} < \iota_i \text{ and } \iota_i > \iota_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

GPR3a (register) recognizes a boundary if the pitch difference between n_2 and n_3 is larger that of n_1 and n_2 and that of n_3 and n_4. GPR3a is formalized as follows.

$$D_{\mathrm{GPR3a}}(i) = \begin{cases} 1 & \text{if } |\eta_{i-1}| < |\eta_i| \text{ and } |\eta_i| > |\eta_{i+1}|, \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

GPR3b (dynamics) recognizes a boundary if there is a change in dynamics between n_2 and n_3, but no changes between n_1 and n_2 nor between n_3 and n_4.

$$D_{\mathrm{GPR3b}}(i) = \begin{cases} 1 & \text{if } \delta_{i-1} = 0, \delta_i \neq 0, \text{ and } \delta_{i+1} = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

GPR3c (articulation) recognizes a boundary if there is a change in articulation between n_2 and n_3, but no changes between n_1 and n_2 nor between n_3 and n_4. Because we could not find a definition of articulation in GTTM (Lerdahl & Jackendoff, 1983), we substitute it with the ratio between the formal length on the score and the real duration in performance. Thus, the definition becomes as follows.

$$D_{\mathrm{GPR3c}}(i) = \begin{cases} 1 & \text{if } \alpha_{i-1} = 0, \alpha_i \neq 0, \text{ and } \alpha_{i+1} = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

GPR3d (length) recognizes a boundary if the duration of n_2 and n_3 are the same while that of n_1 and n_2 and that of n_3 and n_4 are different. GPR3d is formalized as follows.

$$D_{\mathrm{GPR3d}}(i) = \begin{cases} 1 & \text{if } \beta_{i-1} = 0, \beta_i \neq 0, \text{ and } \beta_{i+1} = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{12}$$

GPR4 (intensification) recognizes a boundary if the results by GPR2 and 3 are salient. To formalize GPR4, we consider whether $P_\rho(i)$, $P_\iota(i)$, $P_\eta(i)$, $P_\delta(i)$, $P_\alpha(i)$, $P_\beta(i)$ have larger values, that is, the results of GPR2a, 2b, 3a, 3b, 3c, 3d are salient. T_{GPR4} $(0 \leq T_{\mathrm{GPR4}} \leq 1)$ is the threshold for judging whether GPR4 is applicable, i.e.

whether the effects of GPR2a, 2b, 3a, 3b, 3c, 3d are salient.

$$D_{\mathrm{GPR4}}(i) = \begin{cases} 1 & \text{if } \max(P_\rho(i), P_\iota(i), P_\eta(i), P_\delta(i), P_\alpha(i), \\ & \quad P_\beta(i)) > T_{\mathrm{GPR4}} \\ 0 & \text{otherwise,} \end{cases} \tag{13}$$

where

$$P_\rho(i) = \begin{cases} \rho_i/(\rho_{i-1}+\rho_i+\rho_{i+1}) & \text{if } \rho_{i-1}+\rho_i+\rho_{i+1} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$P_\iota(i) = \iota_i/(\iota_{i-1}+\iota_i+\iota_{i+1}),$$

$$P_\eta(i) = \begin{cases} |\eta_i|/(|\eta_{i-1}|+|\eta_i|+|\eta_{i+1}|) & \text{if } |\eta_{i-1}|+|\eta_i| \\ & \quad +|\eta_{i+1}| > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$P_\delta(i) = \begin{cases} \delta_i/(\delta_{i-1}+\delta_i+\delta_{i+1}) & \text{if } \delta_{i-1}+\delta_i+\delta_{i+1} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$P_\alpha(i) = \begin{cases} \alpha_i/(\alpha_{i-1}+\alpha_i+\alpha_{i+1}) & \text{if } \alpha_{i-1}+\alpha_i+\alpha_{i+1} > 0, \\ 0 & \text{otherwise,} \end{cases}$$

$$P_\beta(i) = \begin{cases} \beta_i/(\beta_{i-1}+\beta_i+\beta_{i+1}) & \text{if } \beta_{i-1}+\beta_i+\beta_{i+1} > 0. \\ 0 & \text{otherwise.} \end{cases}$$

4.1.3 Application of GPR5 (symmetry)

GPR5 prefers that a group be divided into two subgroups of the same length. Otherwise, the more similar the lengths of the two subgroups, the more preferable. In this study, we consider a function that gives priority to the symmetry of the two consecutive subgroups. As a representative of such functions, we employ a Gaussian distribution $D_{\mathrm{GPR5}}(i)$ with the standard deviation σ, the unit of which is a quarter note.

Figure 8(a) shows the distribution $D_{\mathrm{GPR5}}(i)$ of the probability of symmetry. After all of the GPRs are applied, the distribution in the lower grouping level becomes that in Figure 8(b). The nearer the standard deviation σ comes to 0, the narrower the skirt becomes, and the larger the value of $D_{\mathrm{GPR5}}(i)$ becomes. Thus, the mid-point of the upper group tends to be a boundary of subgroups. Conversely, when σ is large, the boundaries of lower groups may deviate from the center of their higher group.

$$D_{\mathrm{GPR5}}(i) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left\{ \frac{-(\tau_i - \tau_{\mathrm{mid}})^2}{2\sigma^2} \right\}, \tag{14}$$

where

$$\tau_{\mathrm{mid}} = \frac{\varepsilon_{\mathrm{end}} - \tau_{\mathrm{start}}}{2},$$

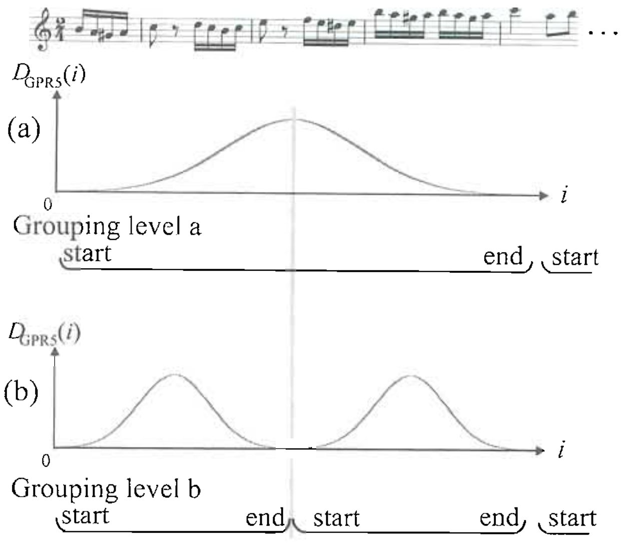


Fig. 8. Degree of symmetry $D_{\mathrm{GPR5}}(i)$.

and "start" is the first note of a group at the current grouping level and "end" is the ending note. "start" and "end" are renewed and new $D_{\mathrm{GPR5}}(i)$ is calculated when the upper group is divided and its lower groups are generated.

The actual adjustable parameter of the system is not σ but $\hat{\sigma} \times (\varepsilon_{\mathrm{end}} - \tau_{\mathrm{start}}) = \sigma$; thus, the skirt of the Gaussian distribution becomes narrower in the lower grouping levels.

4.1.4 Application of GPR6 (parallelism)

GPR6 concerns parallelism, that is, the melodic identity or similarity. Namely, if multiple parts in a musical piece include similar melodies, then the grouping structure should also reflect the same parallelism. Thus, when similar parts are found, both ends of the melody are highly probable to be boundaries. In this section, we discuss the parameter $D_{\mathrm{GPR6}}(i)$ $(0 \leq D_{\mathrm{GPR6}}(i) \leq 1)$, that is, the probability for the note i to be the beginning or the ending note of a parallel phrase. As a result, we expect that the highly probable positions of boundaries are clarified as in Figure 9.

$D_{\mathrm{GPR6}}(i)$ is calculated as follows. First, we calculate the similarity between the interval from note i with length r and the one from j with the same length. Next, for such i, we calculate the similarity for all j's with the same length r. Because there is no description of the similarity of melodies in GTTM (Lerdahl & Jackendoff, 1983), we define our own similarity. Our adjustable parameters for this similarity include:

W_m $(0 \leq W_m \leq 1)$	For each note, give a preference of the similarity of onset time to that of pitch.
W_l $(0 \leq W_l \leq 1)$	Give a preference of the longer interval to the shorter one when

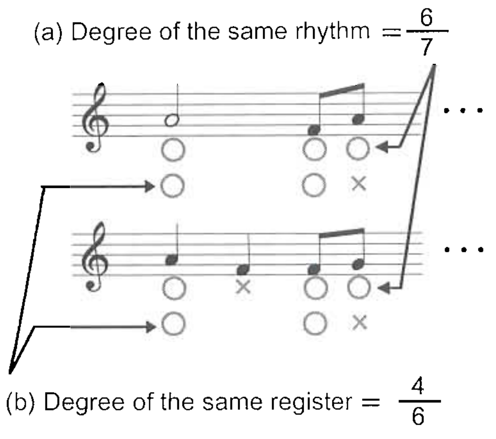

(a) Degree of the same rhythm $= \frac{6}{7}$
(b) Degree of the same register $= \frac{4}{6}$

Fig. 9. Similarity of parallel phrases.

	parallel intervals overlap each other.
W_s $(0 \leq W_s \leq 1)$	For each note, give a preference of being the beginning note of a group to being the ending note.

The similarity in this definition does not affect other parts of the system. Thus, we can substitute other methods (e.g. Hewlett, 1998) for our definition.

Let us first consider the example in Figure 9. In the figure, three notes out of four coincide with regard to onset time. Two notes out of the three also coincide with regard to pitch. In our implementation, we regard a greater number of notes having the same onset time as indicating greater similarity of melodies. Furthermore, the more the number of notes of the same onset time have the same pitch, the more similar the melodies are.

We formalize the above discussion as follows. We assume that the beginning and the ending notes of a group possess beats and that the length of an interval r is a multiple of a beat. Also, we assume that parallelism cannot occur at such adjacent positions, say, as the distance of less than a quarter of a beat. Given a beat number m (≥ 1), we write the interval of r beats from m as $[m, m+r)$, which does not include the $(m+r)$th. We define the basic parameters as follows.

$N(m, r)$	the number of notes in $[m, m+r)$.
$O(m, n, r)$	the number of notes of the same onset time in $(m, m+r)$ and $(n, n+r)$.
$P(m, n, r)$	the number of notes of the same pitch, as well as of the same onset time.

We define the similarity between the interval $(m, m+r)$ and $(n, n+r)$ with these parameters:

$$G(m,n,r) = \left\{ \frac{O(m,n,r)}{N(m,r)+N(n,r)} \times (1 - W_m) + \frac{P(m,n,r)}{O(m,n,r)} \times W_m \right\} \times r^{W_l}, \tag{15}$$

where given the whole number of beats L, $1 \leq m$, $n \leq L - r + 1$ and $1 \leq r \leq L$. Beyond this domain, we regard $G(m, n, r) = 0$.

Note that r^{W_l} becomes 1 when $W_l = 0$, and as r increases, r^{W_l} also increases as far as $W_l > 0$. Thus, as W_l comes close to 1, the similarity of longer intervals becomes significant.

However, there are also meaningless intervals for calculating the similarity. We have assumed that the beginning and the ending notes of an interval must be on certain beats, and thus, those intervals that do not satisfy this condition should be excluded. We introduce a predicate to determine whether the ith note possesses a beat.

$b(i)$	ith note is the beginning one of an interval.
$e(i)$	ith note is the ending one of an interval.
$t(i)$	ith note is either the beginning or ending one of an interval.

We employ the following definitions.

$head(m)$	returns the first note i in $[m, m+r)$.
$tail(m)$	returns the last note i in $[m, m+1)$.
$beat(i)$	returns m if i is in $[m, m+1)$.

Then,

$b(i)$ is true if $i = head(beat(i))$ and $i \neq tail(beat(i))$.
$e(i)$ is true if $i \neq head(beat(i))$ and $i = tail(beat(i))$.
$t(i)$ is true if $i = head(beat(i))$ and $i = tail(beat(i))$.

Next, we calculate the similarity of intervals beginning from the ith note.

$$A(i) = \sum_{n=1}^{L} \sum_{r=1}^{L/2} \begin{cases} G(beat(i), n, r) \times (1 - W_s) \\ \quad \text{if } b(i) \text{ holds and } N(n,1) \geq 1, \\ G(beat(i) - r, n - r, r) \times W_s \\ \quad \text{if } e(i) \text{ holds and } N(n,1) \geq 1, \\ G(beat(i), n, r) \times (1 - W_s) \\ \quad + G(beat(i) - r, n - r, r) \times W_s \\ \quad \text{if } t(i) \text{ holds and } N(n,1) \geq 1, \\ 0 \quad \text{otherwise.} \end{cases} \tag{16}$$

Because parallel intervals do not overlap each other, the length of an interval must be shorter than half of the whole number of beats L. As the final step, we normalize $A(i)$. Let $A_{\max}$ be the maximum value among the notes included in the piece, that is, $A_{\max} = \max(A(1), A(2), \ldots, A(N(1,L))$. Then,

$$D_{\text{GPR6}}(i) = A(i)/A_{\max}. \tag{17}$$

Note that the value of $D_{\text{GPR6}}(i)$ is indifferent to the length of the entire piece. The greater the number of parallel sequences beginning, or ending, with i, the larger the value of $D_{\text{GPR6}}(i)$ is.

In Figure 10, the uppermost graph denotes the possibility that each note can be the beginning of a parallel interval. Similarly, the middle graph shows the possibility of the ending. The bottom graph is the result of combining the above two, together with the normalization.

4.1.5 Application of GPR1 (alternative form)

GPR1 prohibits one note from forming a group. GPR1 gives a criterion whether the ith transition can be a boundary ($D_{\text{GPR1}}(i) = 1$) or not ($D_{\text{GPR1}}(i) = 0$). In Figure 11, $B^{\text{low}}(i)$ represents the local strength by a real number between 0 and 1; the larger the value is, the more likely the boundary is. $D_{\text{GPR1}}(i)$ is calculated in the following way.

$$D_{\text{GPR1}}(i) = \begin{cases} 1 & \text{if } B^{\text{low}}(i-1) \leq B^{\text{low}}(i), \\ & B^{\text{low}}(i) \geq B^{\text{low}}(i+1), \\ & \text{and } D_{\text{GPR1}}(i-1) = 0, \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

where

$$B^{\text{low}}(i) = \frac{\sum_R D_{\text{GPR}_R}(i) \times S_{\text{GPR}_R}}{\max\limits_{i'} \left(\sum_R D_{\text{GPR}_R}(i') \times S_{\text{GPR}_R}\right)} \tag{19}$$

and $R \in \{2a, 2b, 3a, 3b, 3c, 3d, \textit{and } 6\}$.

4.2 Acquisition of hierarchical grouping structure

Local boundaries are detected by $D_{\text{GPR1}}(i)$, $D_{\text{GPR2a}}(i)$, $D_{\text{GPR2b}}(i)$, $D_{\text{GPR3a}}(i)$, $D_{\text{GPR3b}}(i)$, $D_{\text{GPR3c}}(i)$, $D_{\text{GPR3d}}(i)$, and $D_{\text{GPR6}}(i)$. T^{low} is the threshold for determining whether the ith transition becomes a boundary ($D^{\text{low}}(i) = 1$) or not ($D^{\text{low}}(i) = 0$). $B^{\text{low}}(i)$ is defined in (19).

$$D^{\text{low}}(i) = \begin{cases} 1 & \text{if } B^{\text{low}}(i) > T^{\text{low}}, \\ & \text{and } D_{\text{GPR1}}(i) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{20}$$

The hierarchical grouping structure is formed from the local boundaries $D^{\text{low}}(i)$ in the bottom-up way and also from $D_{\text{GPR1}}(i)$, $D_{\text{GPR2a}}(i)$, $D_{\text{GPR2b}}(i)$, $D_{\text{GPR3a}}(i)$, $D_{\text{GPR3b}}(i)$, $D_{\text{GPR3c}}(i)$, $D_{\text{GPR3d}}(i)$, $D_{\text{GPR4}}(i)$, $D_{\text{GPR5}}(i)$, and $D_{\text{GPR6}}(i)$ in the top-down way. $B^{\text{high}}(i)$ represents the strength of a boundary in a higher hierarchy by a real number from 0 to 1. $B^{\text{high}}(i)$ is different from $B^{\text{low}}(i)$ in that the former reflects the result of those rules that concern phrasal structure, that is, $D_{\text{GPR4}}(i)$ and $D_{\text{GPR5}}(i)$. When a group includes a local boundary in it, the boundaries of one upper level $\hat{i}$ is recursively detected by the following procedure (Figure 12), where $B^{\text{high}}(i)$ is renewed at each level, since the value of $D_{\text{GPR5}}(i)$ will change at every grouping level. Then we have a boundary

$$\hat{i} = \operatorname*{argmax}_{i} D^{\text{high}}(i), \tag{21}$$

where

$$D^{\text{high}}(i) = D^{\text{low}}(i) \times B^{\text{high}}(i), \tag{22}$$

$$B^{\text{high}}(i) = \frac{\sum_R D_{\text{GPR}_R}(i) \times S_{\text{GPR}_R}}{\max\limits_{i'} \left(\sum_R D_{\text{GPR}_R}(i') \times S_{\text{GPR}_R}\right)} \tag{23}$$

for $R \in \{2a, 2b, 3a, 3b, 3c, 3d, 4, 5, \textit{and } 6\}$ and for all the i's included in the group.

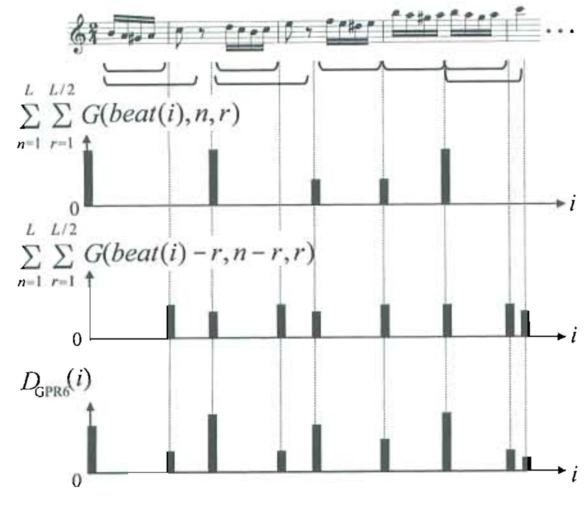


Fig. 10. Degree of parallelism $D_{\text{GPR6}}(i)$.

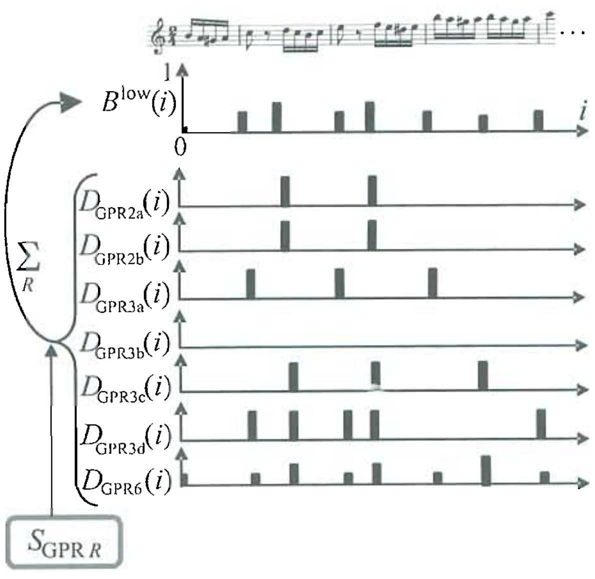
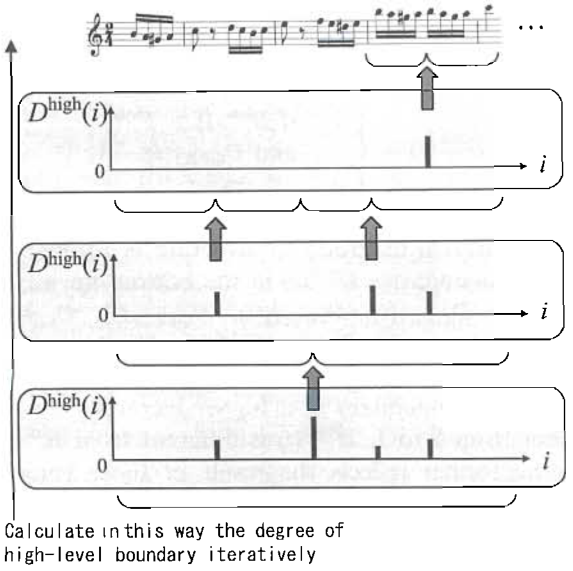


Fig. 11. Low-level strength of boundary $B^{\text{low}}(i)$.



$D^{\text{high}}(i)$
0
i
Calculate in this way the degree of high-level boundary iteratively

Fig. 12. Construction of hierarchical grouping structure.

See Figure 7 for the full procedure of parameter calculation.

5. Metrical analysis

The algorithm of metrical analysis consists of the following steps.

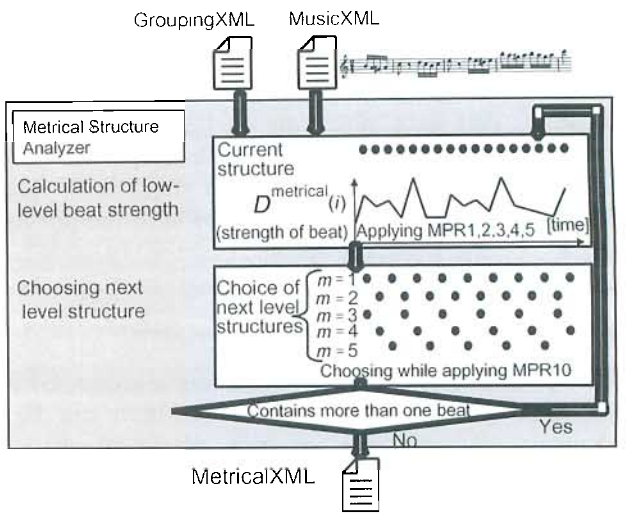
(1) Regard all of the beats in the entire piece as a lowest-level (global) metrical structure.
(2) Apply those rules that are applicable to local metrical structures.
(3) Calculate local-level metrical strength.
(4) Select the strongest metrical structure from the possible structures.
(5) Repeat steps 2 to 4 as long as the current structure contains more than one beat.

According to this procedure, we can construct a hierarchical metrical structure.

Figure 13 shows the processing flow of the metrical analysis. As the primary input formats, we adopt MusicXML (Recordare, 2007a) and GroupingXML (Hamanaka et al., 2004). A hierarchical metrical structure is constructed in a top-down way, while the lower-level beat strength is detected in a bottom-up way. We then design MetricalXML as the export format for our system.

5.1 Application of MPRs

Here, we discuss the applications of MPR1, 2, 3, 4, 5, and 10. We do not include MPR9 because this rule requires sending the information to the later process of time-span



GroupingXML
MusicXML
Metrical Structure Analyzer
Calculation of low-level beat strength
Current structure
$D^{\text{metrical}}(i)$
(strength of beat)
Applying MPR1,2,3,4,5
[time]
Choosing next level structure
Choice of next level structures
$m=1$
$m=2$
$m=3$
$m=4$
$m=5$
Choosing while applying MPR10
Contains more than one beat
Yes
No
MetricalXML

Fig. 13. Processing flow of metrical-structure analyser.

reduction and back to the earlier process of metrical analysis. Furthermore, we do not deal with MPR6, MPR7, or MPR8 because we restrict our target music to monophony. The strength of the beat dependent on each rule can be expressed by $D_{\text{MPR}_R}(i)$ ($R \in \{1, 2, 3, 4, 5\text{a}, 5\text{b}, 5\text{c}, 5\text{d}, \textit{and } 5\text{e}\}$), where $0 \leq D_{\text{MPR}_R}(i) \leq 1$. As there is no strict order in applying MPR rules, ambiguities may result from the analysis. To solve this problem, we use adjustable parameters $S_{\text{MPR}_R}(i)$ ($R \in \{1, 2, 3, 4, 5\text{a}, 5\text{b}, 5\text{c}, 5\text{d}, 5\text{e}, \textit{and } 10\}$), which enable us to control the strength of each rule. Figure 14 shows the relationship between the parameters and MPRs. Our metrical-structure analyser has 18 adjustable parameters, which include S_{MPR_R}, $\acute{W}_m$, $\acute{W}_r$, $\acute{W}_l$, and T_{MPR_R} (Table 2).

5.1.1 Calculation of basic parameters

From the input MusicXML, the following five basic parameters of the ith beat in the current hierarchy of beats are calculated. When there is no note in the position of the ith beat, each basic parameter makes zero.

γ_i	dynamic
λ_i	length of note
ζ_i	duration of dynamic
κ_i	length of slur
ν_i	pitch

Since GTTM (Lerdahl & Jackendoff, 1983) does not include the notion of 'duration of dynamic,' we define it as the length from one beat to the next beat or rest. The averages of the above basic parameters are denoted as $\bar{\gamma}, \bar{\lambda}, \bar{\zeta}, \bar{\kappa}$, and $\bar{\nu}$, respectively. From the result of GroupingXML, the two basic parameters i_s and i_e are the beginning and the ending, respectively, of the smallest group containing i and more than one beat in the current structure.

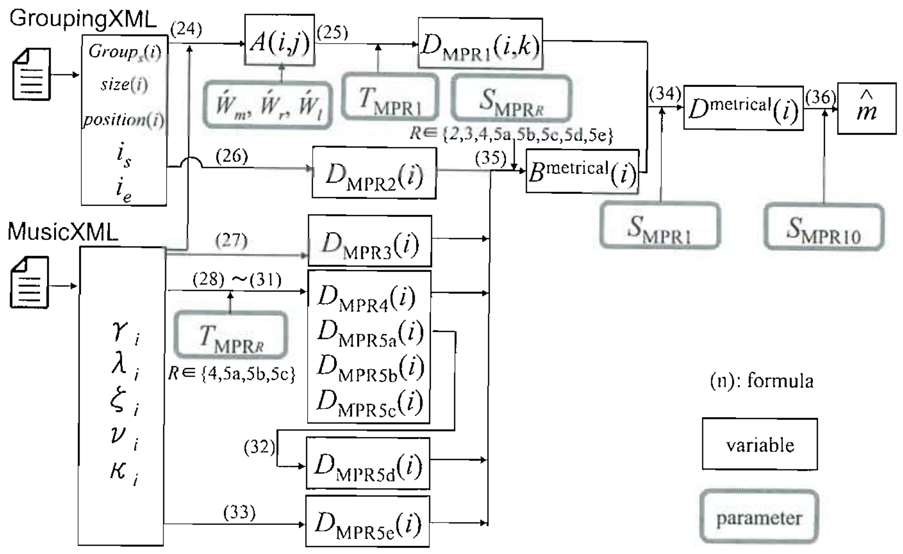


Fig. 14. Parameters in MPRs.

5.1.2 Application of MPR1 (parallelism)

MPR1 concerns the parallelism in metrical structures and prefers that a pair of parallel groups have similar metric structures. The similarity is evaluated basically in the same way as done in GPR6 (15), but the intervals to be estimated are different. In the case of GPR6, we have targeted all of the intervals with length r, while in MPR1 we apply the method only to the resultant groups of the grouping analysis. Then, the degree of parallelism $A(i, j)$ between the ith beat in the interval $(m, m+r)$ and the jth beat in $(m, m+s)$ becomes

$$A(i,j) = G(group_s(i), group_s(j), size(i)), \qquad (24)$$

where $group_s(i)$ is the first beat of the group including i, and $size(i)$ is the number of beats of the group. When the value of $A(i, j)$ goes beyond the threshold T^{MPR1} after normalization, we judge that the ith beat and jth beat are parallel.

$$D_{\mathrm{MPR1}}(i,j) = \begin{cases} 1 & A(i,j)/A_{\max} > T^{\mathrm{MPR1}}, \\ & \text{and } position(i) = position(j), \\ 0 & \text{otherwise,} \end{cases} \qquad (25)$$

given

$$A_{\max} = \max\left(A(i,1), A(i,2), \ldots, A(i,L)\right),$$

where $position(i)$ is the temporal duration from the beginning of the group including the i-beat to the ith beat itself.

5.1.3 Application of MPR2 and 3

MPR2 (strong beat early) has a weak preference for a metrical structure in which the strongest beat in a group appears relatively early in the group. We formalized $D_{\mathrm{MPR2}}(i)$ so that the closer the strongest beat appears to the beginning of the group, the higher the value is

$$D_{\mathrm{MPR2}}(i) = (i_e - i)/(i_e - i_s). \qquad (26)$$

MPR3 (event) prefers a metrical structure in which the inceptions of pitch-events are on strong beats (Figure 15). Thus, $D_{\mathrm{MPR3}}(i)$ returns 1 if i has a beat, and 0 otherwise.

$$D_{\mathrm{MPR3}}(i) = \begin{cases} 1 & \gamma_i > 0, \\ 0 & \gamma_i = 0. \end{cases} \qquad (27)$$

5.1.4 Application of MPR4, 5a, 5b, and 5c

We introduced the adjustable parameters (T_{MPR_R} ($R \in \{4, 5a, 5b, \text{and } 5c\}$) ($0 \le T_{\mathrm{MPR}_R} \le 1$) to control the threshold that determines whether each rule is applicable, i.e. ($D_{\mathrm{MPR}_R}(i) = 1$) or not ($D_{\mathrm{MPR}_R}(i) = 0$) (Figure 16).

MPR4 (stress) prefers a metrical structure in which a relatively strong beat occurs at the inception of a stressed note. $D_{\mathrm{MPR4}}(i)$ returns 1 if a stressed note has a strong beat, and 0 otherwise.

$$D_{\mathrm{MPR4}}(i) = \begin{cases} 1 & \gamma_i > 2 \times \bar{\gamma} \times T_{\mathrm{MPR4}}, \\ 0 & \text{otherwise.} \end{cases} \qquad (28)$$

MPR5a (long pitch-event) prefers a metrical structure in which a relatively strong beat occurs at the inception of a long note. $D_{\mathrm{MPR5a}}(i)$ returns 1 if a relatively longer note possesses a strong beat, and 0 otherwise.

$$D_{\mathrm{MPR5a}}(i) = \begin{cases} 1 & \lambda_i > 2 \times \bar{\lambda} \times T_{\mathrm{MPR5a}}, \\ 0 & \text{otherwise.} \end{cases} \qquad (29)$$

MPR5b (long duration of a dynamic) prefers a metrical structure in which a relatively strong beat occurs at the

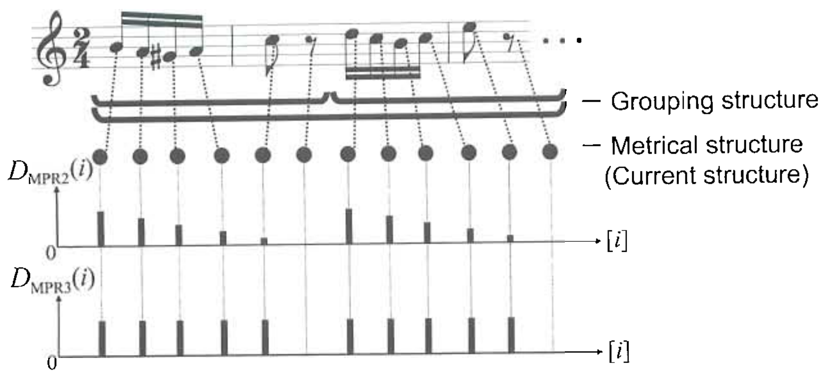
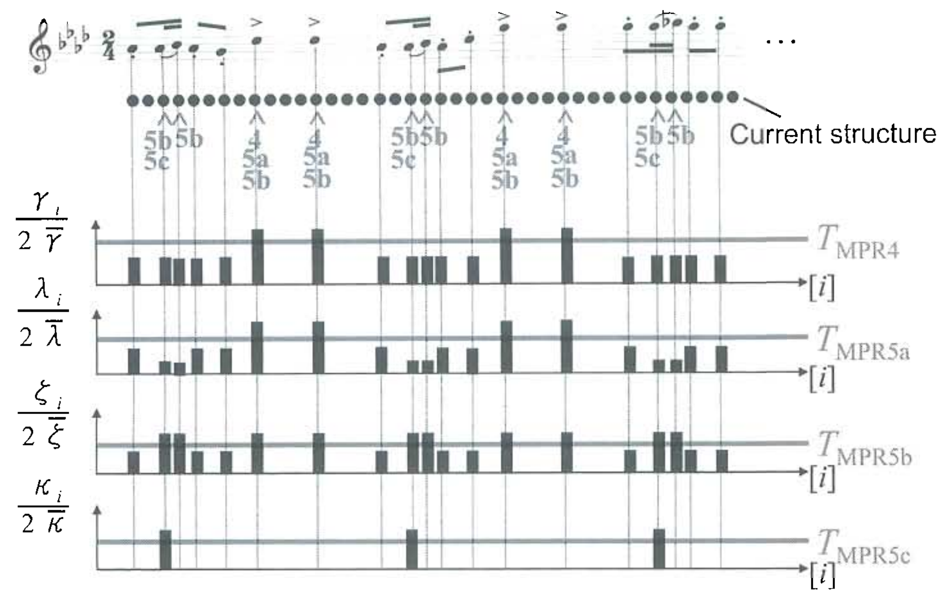


Fig. 15. Application of MPR2 and 3.



Current structure
5b 5b 5c
4 5a 5b
$\frac{\gamma_i}{2\bar{\gamma}}$
$\frac{\lambda_i}{2\bar{\lambda}}$
$\frac{\zeta_i}{2\bar{\zeta}}$
$\frac{\kappa_i}{2\bar{\kappa}}$
T_{MPR4}
T_{MPR5a}
T_{MPR5b}
T_{MPR5c}
$[i]$

Fig. 16. Application of MPR4, 5a, 5b and 5c.

inception of a long duration of a dynamic. $D_{\mathrm{MPR5b}}(i)$ returns 1 if a dynamically large note possesses a strong beat, and 0 otherwise.

$$D_{\mathrm{MPR5b}}(i) = \begin{cases} 1 & \zeta_i > 2 \times \bar{\zeta} \times T_{\mathrm{MPR5b}}, \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

MPR5c (long slur) prefers a metrical structure in which a relatively strong beat occurs at the inception of a long slur. $D_{\mathrm{MPR5c}}(i)$ returns 1 if a relatively long slur possesses a strong beat, and 0 otherwise.

$$D_{\mathrm{MPR5c}}(i) = \begin{cases} 1 & \kappa_i > 2 \times \bar{\kappa} \times T_{\mathrm{MPR5c}}, \\ 0 & \text{otherwise.} \end{cases} \tag{31}$$

5.1.5 Application of MPR5d and 5e

MPR5d (long pattern of articulation) prefers that the relatively long repetition of the same articulation have a strong beat. Because we could not find a definition for this *repetition*, we regard it as the repetitive application of MPR5a according to the examples in GTTM (Lerdahl & Jackendoff, 1983). $D_{\mathrm{MPR5d}}(i)$ returns 1 if MPR5a is consecutively applied, and 0 otherwise.

$$D_{\mathrm{MPR5d}}(i) = \begin{cases} 1 & D_{\mathrm{MPR5a}}(i) = 1 \text{ and } D_{\mathrm{MPR5a}}(i+1) = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{32}$$

MPR5e (long duration of a pitch) is the rule for pitch-repetition. $D_{\mathrm{MPR5e}}(i)$ returns 1 if the same pitch persists, and 0 otherwise.

$$D_{\mathrm{MPR5e}}(i) = \begin{cases} 1 & \nu_i = \nu_{i+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{33}$$

5.2 Acquisition of hierarchical metrical structure

Low-level beat strength $D^{\mathrm{metrical}}(i)$ is calculated by $D_{\mathrm{MPR1}}(i, j)$, $D_{\mathrm{MPR2}}(i)$, $D_{\mathrm{MPR3}}(i)$, $D_{\mathrm{MPR4}}(i)$,

$D_{\mathrm{MPR5a}}(i)$, $D_{\mathrm{MPR5b}}(i)$, $D_{\mathrm{MPR5c}}(i)$, $D_{\mathrm{MPR5d}}(i)$, and $D_{\mathrm{MPR5e}}(i)$.

$$D^{\mathrm{metrical}}(i) = B^{\mathrm{metrical}}(i) + \sum_j (B^{\mathrm{metrical}}(j) \times S_{\mathrm{MPR1}} \text{ if } D_{\mathrm{MPR1}}(i,j) = 1), \tag{34}$$

where

$$B^{\mathrm{metrical}}(i) = \sum_R D_{\mathrm{MPR}_R}(i) \times S_{\mathrm{MPR}_R} \tag{35}$$

and $R \in \{2, 3, 4, 5a, 5b, 5c, \mathit{and}\ 5d\}$. $B^{\mathrm{metrical}}(i)$ represents the weighted summation of S_{MPR_R} and $D_{\mathrm{MPR}_R}(i)$, ($R \in 2$, 3, 4, 5a, 5b, 5c, 5d, *and* 5e), and $D^{\mathrm{metrical}}(i)$ represents the sum of $B^{\mathrm{metrical}}(i)$ and summation of $B^{\mathrm{metrical}}(j)$, where the ith beat and jth beat are parallel and consequently $D_{\mathrm{MPR1}}(i, j) = 1$.

A hierarchical metrical structure is constructed by iterating the calculation of the low-level strength of the beat $D^{\mathrm{metrical}}(i)$ for the current structure and choosing the next-level structure. When the current structure contains more than one beat, the next-level structure is chosen as follows. First, we assume that the piece is in simple duple time; then we compare among

$$\left\{ \sum_{\{i|i=2k-1\}} D^{\mathrm{metrical}}(i), \sum_{\{i|i=2k\}} D^{\mathrm{metrical}}(i) \right\} \tag{36}$$

($k = 1, 2, 3, \ldots$), and choose the larger one, i.e. either the odd i's or the even i's would organize the upper structure. In a similar way, when we assume that the piece is in simple triple time, i's are collected in the three different ways: $\{i|i = 3k - 2\}$, $\{i|i = 3k - 1\}$, or $\{i|i = 3k\}$ ($k = 1, 2, 3, \ldots$), and we choose the set by which

$$\sum_i D^{\mathrm{metrical}}(i) \times S_{\mathrm{MPR10}} \tag{37}$$

becomes largest. Note that **MPR10** prefers metrical structures in which every other beat is strong so that the simple triple time is deterred by S_{MPR10}.

In Figure 17, $\hat{m}$ shows the choice of

$$\hat{m} = \begin{cases} 1 & \mathit{iff} \quad i = \{1,3,5,\ldots\} \\ 2 & \mathit{iff} \quad i = \{2,4,6,\ldots\} \\ 3 & \mathit{iff} \quad i = \{1,4,7,\ldots\} \\ 4 & \mathit{iff} \quad i = \{2,5,8,\ldots\} \\ 5 & \mathit{iff} \quad i = \{3,6,9,\ldots\} \end{cases} \tag{38}$$

including both of simple duple and simple triple time.

As a result, each note or rest of a music piece is marked by a *dot* if it possesses a strong beat at the certain level of the hierarchy. According to the difference in levels and the strength of a beat, each note has a different number of dots.

6. Generation of time-span tree

The time-span tree is constructed in the following way.

(1) Consider all of the notes as a head.
(2) Apply those rules that are applicable to local-level heads.
(3) Calculate the head strength at a local-level.
(4) Select the next-level head from each time-span.
(5) Iterate steps 2 and 4 as long as the time-span contains more than one head.

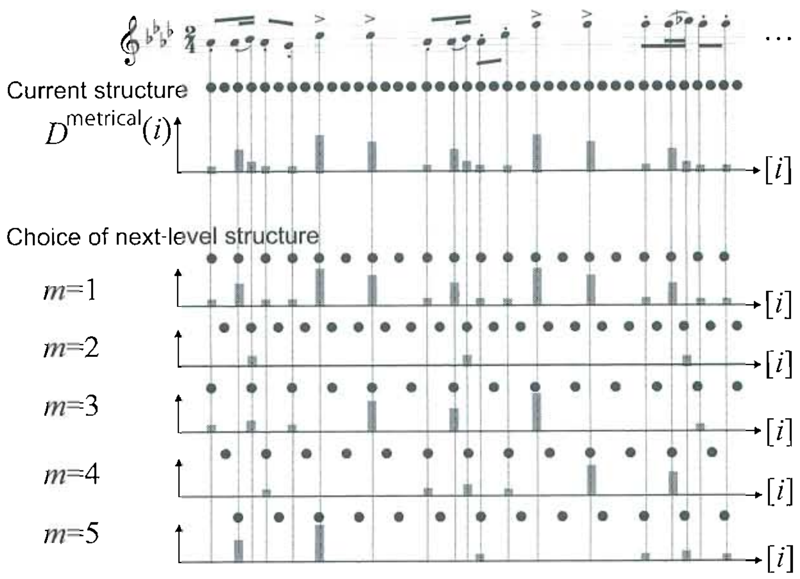


Fig. 17. Selecting the next-level structure.

Here, we employ the adjustable parameters S_{TSRPR_R} ($R \in \{1, 3a, 3b, 4, 8, \mathit{and}\, 9\}$), each of which is the strength of a time-span tree preference rule. Figure 18 shows the processing flow of the time-span reduction. As the input formats, we adopt MusicXML (Recordare, 2007a), GroupingXML (Hamanaka et al., 2004) and MetricalXML (Hamanaka et al., 2005a). We then design Time-spanXML as the export format for our system.

6.1 Application of time-span reduction preference rules

In this section, we explain our application methods of TSRPR$_R$ ($R \in \{1, 3, 4, 8, \mathit{and}\ 9\}$). In this study, we restrict our target music to monophonic music, so we do not deal with TSRPR2 and TSRPR7. Furthermore, we do not implement TSRPR5 and TSRPR6 because they suggest a *reverse* flow of processing, i.e. they select a time-span tree that makes metric structures or prolongational reductions stable. $D_{\mathrm{TSRPR}_R}(i)$ ($R \in \{1, 3a, 3b, 4, 8, \mathit{and}\ 9\}$) indicates whether TSRPR$_R$ holds. Since the priority among these TSRPRs was not shown in GTTM (Lerdahl & Jackendoff, 1983) as were GPRs and MPRs, we introduce adjustable parameters S_{TSRPR_R} ($R \in \{1, 3a, 3b, 4, 8, \mathit{and}\ 9\}$). Figure 19 shows the dependency between the application of TSRPRs. Our time-span tree analyser has 13 adjustable parameters, which include $S_{\mathrm{TSRPR_R}}$, $\acute{W}_m$, $\grave{W}_m$, and $\check{W}_m$ (Table 3).

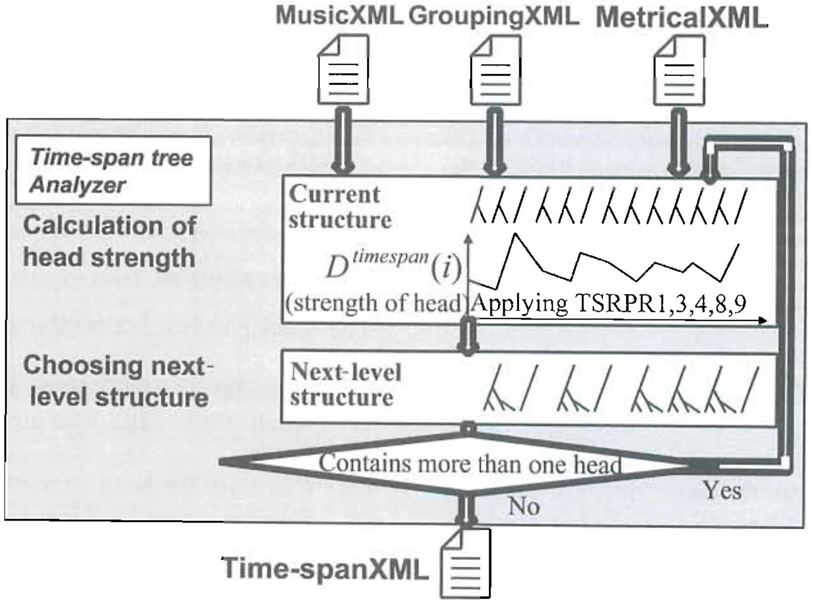


Fig. 18. Processing flow of time-span tree analyser.

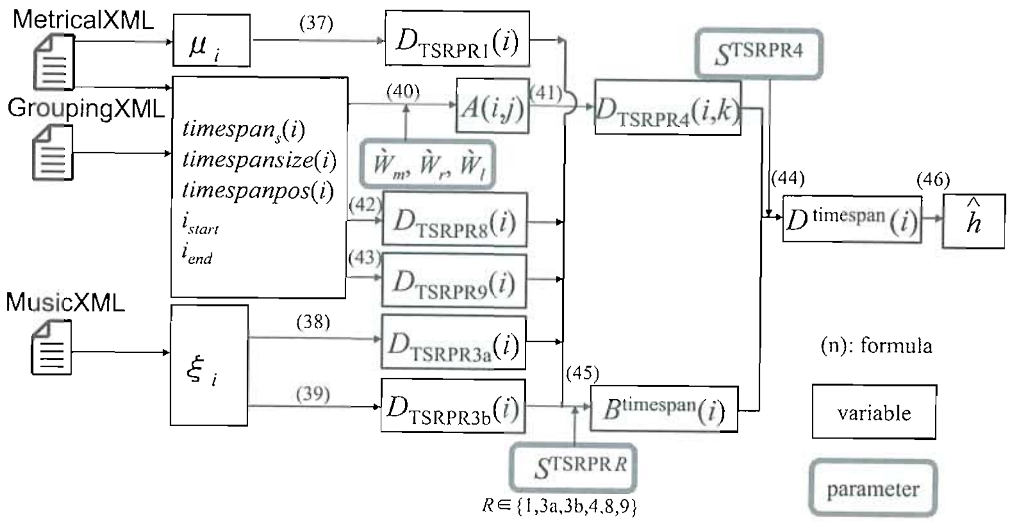


Fig. 19. Parameters in TSRPRs.

6.1.1 Time-span segmentation

In the procedure of time-span reduction, we divide the entire piece into hierarchical time-spans. We show the division procedure in Figure 20, which is:

(1) Regard all of the resultant groups of grouping analysis as time-spans.
(2) When a time-span in the lowest level includes more than one note, divide it into two at the strongest beat.
(3) Repeat 2 recursively.

In the GTTM (Lerdahl & Jackendoff, 1983, pp. 156–157) there are two rules for time-span segmentation, called segmentation rule 1 and segmentation rule 2. Segmentation rule 1 corresponds to the first item and segmentation rule 2 corresponds to the second item.

As a result, a music piece is formed into a binary tree, i.e. a time-span tree, at each node of which the more important branch extends upward as a head note. The selection of a head at each node is hierarchically computed from the lower level. Therefore, heads are selected from leaves to root branches. For each level in the hierarchy, we provide the following basic parameters and the rule application principles.

6.1.2 Calculation of basic parameters

We calculate four basic parameters:

μ_i	Number of dots
ϕ_i	Offset-to-onset Interval (OOI)
ψ_i	Inter-onset interval (IOI)
ξ_i	Difference in pitch

In the first three parameters, i represents the ith gap of heads, that is, the gap between the ith and the $(i+1)$th head while dot_i is the number of dots of the ith head. i indicates the order of heads at the current level of time-span. The basic parameters are renewed at every hierarchy of time-span because the number of heads changes as a result of selecting heads at every hierarchy of time-span.

6.1.3 Application of TSRPR1 (metrical position)

TSRPR1 prefers a head with a stronger beat. We normalize the strength between 0 and 1 and define the likelihood of being a head by the number of dots divided by the maximum number of dots.

$$D_{\text{TSRPR1}}(i) = \mu_i / \max_j \mu_j. \tag{39}$$

6.1.4 Application of TSRPR3 (registral extremes)

TSRPR3 concerns the pitch of a head. TSRPR3a selects a higher pitch note as a head while TSRPR3b prefers a lower one. Thus, $D_{\text{TSRPR3a}}(i)$ returns a higher value if ξ_i is higher.

$$D_{\text{TSRPR3a}}(i) = \xi_i / \max_j \xi_j. \tag{40}$$

Conversely, $D_{\text{TSRPR3b}}(i)$ returns a higher value if ξ_i is lower.

$$D_{\text{TSRPR3b}}(i) = 1 - \xi_i / \max_j \xi_j. \tag{41}$$

6.1.5 Application of TSRPR4 (parallelism)

TSRPR4 concerns the parallelism and prefers a head in a parallel position in parallel time-spans. The parallelism here is different from the melodic similarity: even though two phrases are not similar, they may be regarded as parallel due to the time-span reduction. The parallelism is evaluated among heads in the current hierarchy's

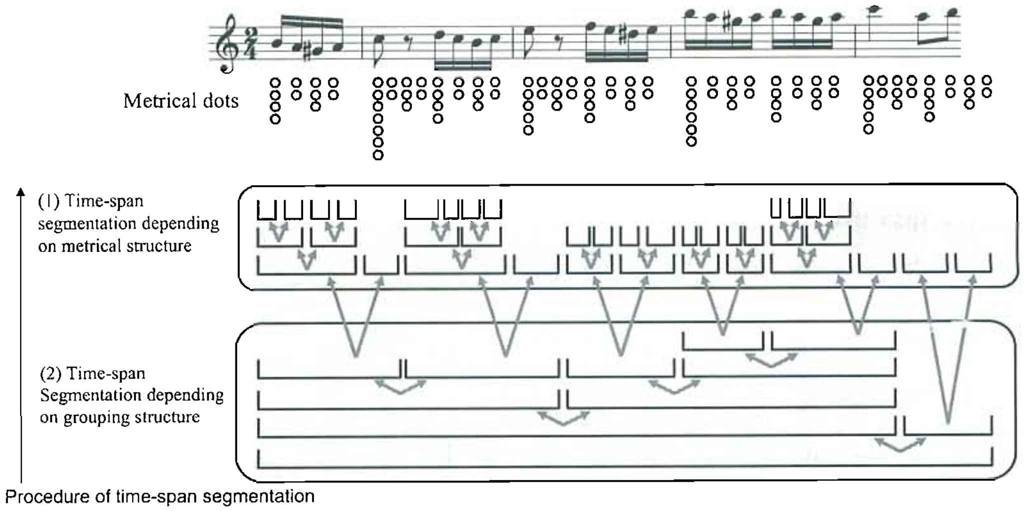

Metrical dots
(1) Time-span segmentation depending on metrical structure
(2) Time-span Segmentation depending on grouping structure
Procedure of time-span segmentation

Fig. 20. Time-span segmentation.

time-spans, and the evaluation method is the same as that in the grouping and metrical analysis. The similarity of head i in $[m, m+r)$ and j in $[m, m+s)$ is shown by

$$A(i,j) = G(timespan_s(i), timespan_s(j), timespansize(i)), \tag{42}$$

where $timespan_s(i)$ is the first beat of the time-span including i, and $timespansize(i)$ is the length (the number of beats) of the time-span including i. We define $D_{\mathrm{TSRPR4}}(i, j)$, normalizing $A(i, j)$:

$$D_{\mathrm{TSRPR4}}(i,j) = \begin{cases} A(i,j)/A_{\max} & timespanpos(i) = timespanpos(j), \\ 0 & \text{otherwise}, \end{cases} \tag{43}$$

where $A_{\max} = \max(A(i, 1), A(i, 2), \ldots, a(i, L))$ and $timespanpos(i)$ is the interval from the inception of the time-span to i. Note that i indicates the order of heads at the current time-span level, and then $D_{\mathrm{TSRPR4}}(i, j)$ renews at each level of time-span.

6.1.6 Application of TSRPR8 (structural beginning)

TSRPR8 prefers that a head i appear at the beginning of the time-span. $D_{\mathrm{TSRPR8}}(i)$ returns 1 if the head is at the beginning position, and 0 otherwise.

$$D_{\mathrm{TSRPR8}}(i) = \begin{cases} 1 & i = i_{\mathrm{start}}, \\ 0 & \text{otherwise}, \end{cases} \tag{44}$$

where i_{start} is the head of the beginning of the time-span.

6.1.7 Application of TSRPR9 (structural ending)

TSRPR9 prefers that a head i appear at the tail of the time-span. $D_{\mathrm{TSRPR9}}(i)$ returns 1 if the head is at the tail position, and 0 otherwise.

$$D_{\mathrm{TSRPR9}}(i) = \begin{cases} 1 & i = i_{\mathrm{end}}, \\ 0 & \text{otherwise}, \end{cases} \tag{45}$$

where i_{end} is the head of the tail of the time-span.

6.2 Generation of time-span tree

We calculate the plausibility of the head $D^{\mathrm{timespan}}(i)$ by $D_{\mathrm{TSRPR1}}(i)$, $D_{\mathrm{TSRPR3a}}(i)$, $D_{\mathrm{TSRPR3b}}(i)$, $D_{\mathrm{TSRPR4}}(i, j)$, $D_{\mathrm{TSRPR8}}(i)$, and $D_{\mathrm{TSRPR9}}(i)$.

$$D^{\mathrm{timespan}}(i) = B^{\mathrm{timespan}}(i) + \sum_k \begin{cases} B^{\mathrm{timespan}}(k) \times S_{\mathrm{TSRPR4}} & \text{if } D_{\mathrm{TSRPR4}}(i,k) = 1, \\ 0 & \text{if } D_{\mathrm{TSRPR4}}(i,k) = 0, \end{cases} \tag{46}$$

where

$$B^{\mathrm{timespan}}(i) = \sum_R D_{\mathrm{TSRPR}_R}(i) \times S_{\mathrm{TSRPR}_R} \tag{47}$$

and $R \in \{1, 3a, 3b, 8, \textit{and } 9\}$. $B^{\mathrm{timespan}}(i)$ represents weighted summation of S_{TSRPR_R} and $D_{\mathrm{TSRPR}_R}(i)$, ($R \in$ 1, 3a, 3b, 8, *and* 9), and $D^{\mathrm{timespan}}(i)$ represents the sum of $B^{\mathrm{timespan}}(i)$ and summation of $B^{\mathrm{timespan}}(k)$, where the ith head and kth head are parallel and consequently $D_{\mathrm{TSRPR4}}(i, k) = 1$.

A hierarchical time-span tree is constructed by iterating the calculation of the plausibility of the head $D^{\mathrm{timespan}}(i)$ for the current heads and choosing the next-level heads (Figure 21).

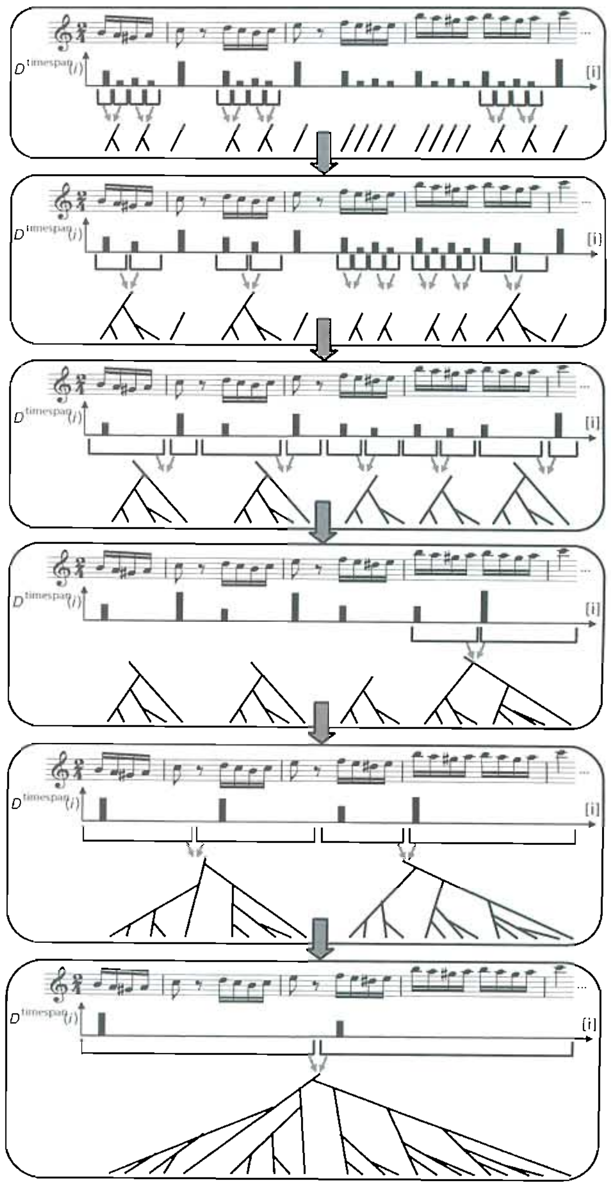


Fig. 21. Selecting the next-level heads.

If there are two candidates of a head in the current hierarchy, we choose the next-level one $\hat{h}$ as 48.

$$\hat{h} = \begin{cases} i & D^{\text{timespan}}(i) \leq D^{\text{timespan}}(j), \\ j & \text{otherwise}, \end{cases} \tag{48}$$

The order of choosing the next-level head is the reverse order of constructing time-spans in the time-span segmentation, as described in Section 6.1.1. $D_{\text{TSRPR}_R}(i)$ and $D^{\text{timespan}}(i)$ are renewed at each level of hierarchy, since i indicate the order of heads in the current hierarchy and changes at each hierarchy of time-spans.

7. ATTA (automatic time-span tree analyser)

We have implemented exGTTM on a computer called ATTA: *Automatic Time-span Tree Analyzer*. ATTA has three distinctive features: an XML-based data structure, implementation in Perl, and a Java-based GUI.

7.1 XML-based data structure

We use an XML format for all of the ATTA's input and output data structures. Each analyser in ATTA works independently but is integrated with the other through the XML-based data structure. As a primary input format, we chose MusicXML (Recordare, 2007a) because it provides a common 'interlingua' for music notation, analysis, retrieval, and other applications. We designed GroupingXML, MetricalXML, and Time-spanXML as the export formats for our analyser (Figure 22). The XML format is very capable of expressing the hierarchical grouping structures, metrical structures, and timespan trees. Note elements in GroupingXML, MetricalXML, and Time-spanXML are connected to note elements in MusicXML by using Xpointer (W3C, 2002) and Xlink (W3C, 2001). We assume that the distribution of a MusicXML or a standard MIDI file, together with a grouping structure, metrical structure, and time-span tree, would be useful for various musical tasks such as searching and arranging.

7.2 Implementation in Perl

We implemented ATTA in Perl, so using CGI allows it to function through the internet (available at http://staff.aist.go.jp/m.hamanaka/atta/). We believe that the availability of this kind of resource is very important for the music-researching community. ATTA is the first application for automatically acquiring a time-span tree. We hope to benchmark ATTA against other systems to be constructed in the future.

Figure 23 shows screen snapshots of the grouping-structure analyser, metrical structure analyser, and time-span tree analyser. The left side of each has scrolling bars for controlling adjustable parameters, and the right side shows the analysis results.

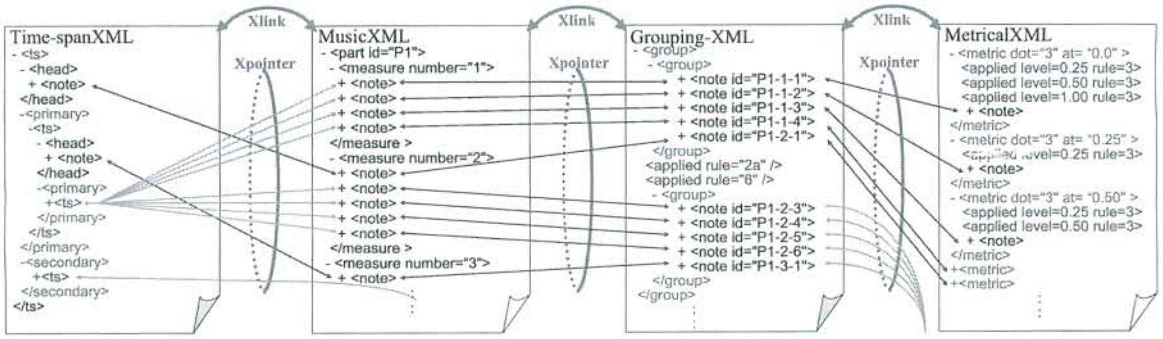
7.3 Java-based GUI

Although our analyser implemented in Perl has a simple user interface, we also developed a graphical user interface in Java called GTTM editor (Figure 24). GTTM editor has two modes, the automatic-analysis and manual-edit modes. The automatic-analysis mode performs analysis by using our analyser and displays the results. The structures change depending on the configured parameters. The manual-edit mode assists in editing the grouping structure, metrical structure, and time-span tree. It can be used to edit the results of the automatic-analysis mode.

8. Experimental results

We evaluated the performance of the music analyser using an F_{measure}, which is given by the weighted harmonic mean of Precision P (proportion of selected groupings/dots/heads that are correct) and Recall R (proportion of correct groupings/dots/heads that were identified).

$$F_{\text{measure}} = 2 \times \frac{P \times R}{P + R}. \tag{49}$$



```
- <ts>
  - <head>
    + <note>
  </head>
  -<primary>
    -<ts>
      - <head>
        + <note>
      </head>
      -<primary>
        +<ts>
      </primary>
    </ts>
  </primary>
  -<secondary>
    +<ts>
  </secondary>
</ts>
```

```
- <part id="P1">
  - <measure number="1">
    + <note>
    + <note>
    + <note>
    + <note>
  </measure >
  - <measure number="2">
    + <note>
    + <note>
    + <note>
    + <note>
    + <note>
  </measure >
  - <measure number="3">
    + <note>
```

```
- <group>
  - <group>
      + <note id="P1-1-1">
      + <note id="P1-1-2">
      + <note id="P1-1-3">
      + <note id="P1-1-4">
      + <note id="P1-2-1">
    </group>
  <applied rule="2a" />
  <applied rule="6" />
   - <group>
      + <note id="P1-2-3">
      + <note id="P1-2-4">
      + <note id="P1-2-5">
      + <note id="P1-2-6">
      + <note id="P1-3-1">
    </group>
</group>
```

```
- <metric dot="3" at= "0.0" >
    <applied level=0.25 rule=3>
    <applied level=0.50 rule=3>
    <applied level=1.00 rule=3>
   + <note>
</metric>
- <metric dot="3" at= "0.25" >
    <applied level=0.25 rule=3>
   + <note>
</metric>
- <metric dot="3" at= "0.50" >
    <applied level=0.25 rule=3>
    <applied level=0.50 rule=3>
   + <note>
</metric>
+<metric>
+<metric>
```

Fig. 22. GroupingXML, MetricalXML, Time-spanXML, and MusicXML.

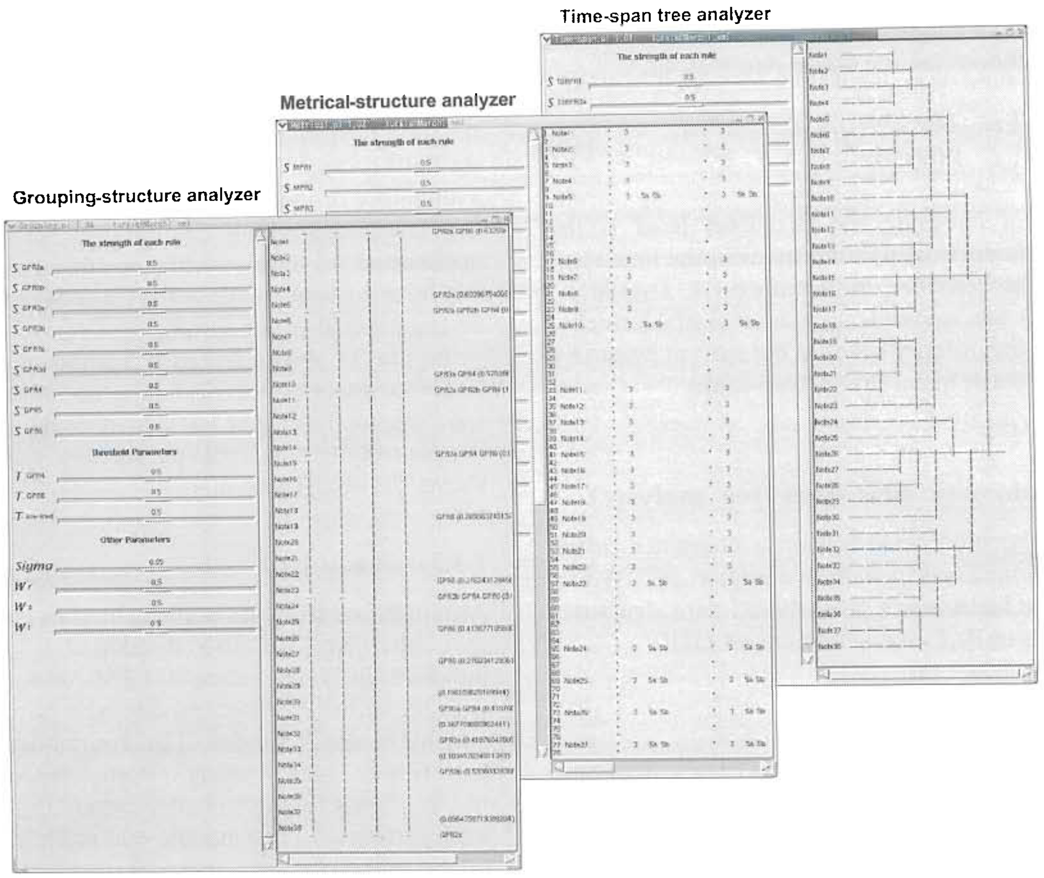

Time-span tree analyzer
Metrical-structure analyzer
Grouping-structure analyzer
The strength of each rule
Threshold Parameters
Other Parameters
Sigma

Fig. 23. Screen snapshots of analysers.

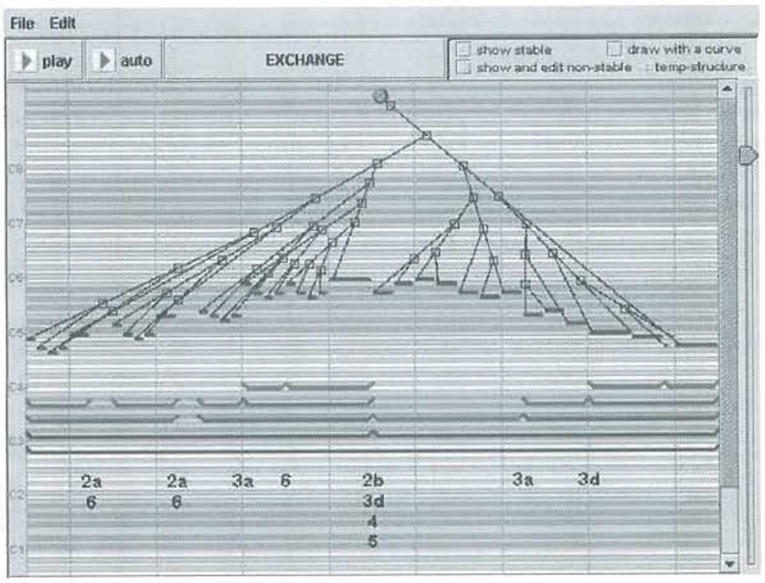


Fig. 24. Screen snapshot of Java-based GUI.

In calculating F_{measure} of the grouping analyser and time-span tree analyser, we did not take into consideration the possibility that a low-level error is propagated up to a higher level; we counted wrong answers without regard to the differences in grouping levels and time-span levels.

8.1 Evaluation data

The evaluation by F_{measure} required us to prepare correct data for the grouping structure, metrical structure, and time-span tree. Because there is no database containing the analysis results by GTTM, we prepared a novel type of evaluation data. The evaluation data include a pair of score data and correct data.

8.1.1 Score data

We collected one hundred 8-bar-long, monophonic, classical music pieces that include notes, rests, slurs, accents, and articulations entered manually with music notation software called *Finale* (PG Music, 2007). In addition, we exported the MusicXML by using a plugin called *Dolet* (Recordare, 2007b).

The basic parameters calculated in Sections 4, 5, and 6 include parameters that cannot be acquired from MusicXML directly. For example, we could not acquire ρ_i from MusicXML directly because its calculation requires real offset times $\hat{\varepsilon}$, but MusicXML only describes the formal offset time ε.

In the experiment, we defined $\hat{\varepsilon}$ so that the duration rate of the formal length of a note and the real length of a note would be 1.0 where the note has a slur, and otherwise it would be 0.8. We also defined the value of velocity υ so that it would be 1.0 where the note has an accent, otherwise it would be 0.8.

8.1.2 Correct data

We asked a musicology expert to manually analyse the score data faithfully with regard to GTTM (Lerdahl & Jackendoff, 1983), using the manual-edit mode of GTTM editor to assist in editing the grouping structure, metrical structure, and time-span tree. Three other experts crosschecked these manually produced results.

8.2 Parameter tuning

The grouping, metrical and time-span tree structures will change depending on the adjustable parameters. To evaluate the baseline performance of our system, we used the following default parameters: $S_{\text{rules}}=0.5$, $T_{\text{rules}}=0.5$, $W_s=0.5$, $W_r=0.5$, $W_l=0.5$, and $\sigma=0.05$. The range of parameters of T_{rules}, W_s, W_r, W_l is 0 to 1.0 and resolution is 0.1. The range of a parameter of σ is 0 to 0.1 and resolution is 0.01.

In the current stage, the parameters are configured by humans because the optimal values of the parameters depend on the piece of music. The experimenter used GTTM editor and a GUI for configuring parameters. When a user changes the parameters, the hierarchical grouping/metrical/time-span tree structures change as a result of the new analysis, and the new results are displayed on GTTM editor. It took us an average of about 10 min per piece to find the plausible tuning for the set of parameters (Tables 1, 2, and 3). As a result of configuring the parameters, each F_{measure} of our analyser outperformed the baseline (Table 4).

8.3 Acquisition of low-level grouping boundary

Here, we discuss whether the low-level grouping boundary was acquired properly. GTTM (Lerdahl & Jackendoff, 1983) presents an analysis of the Mozart Sonata K. 331 as an example that can be interpreted in two ways: as a grouping structure that has a boundary between note 4 and note 5 (figure 25(a)) and as a grouping structure that has a boundary between note 5 and note 6 (figure 25(b)). The system can output both grouping structures properly as a result of using exGTTM for analysis by configuring S_{GPR2a}, S_{GPR2b}, and S_{GPR3a}. Figure 26 shows the analysis results of Chopin, Ballade op. 23, which provide an example of wrong results. The distinctive feature of this musical piece is that a characteristic of the melody changes in the middle of the piece. Figure 26(a) shows the analysis results in which

Table 4. F_{measure} for our method and baseline.

	Grouping Structure		Metrical Structure		Time-Span Tree	
Melodies	Baseline	Our method	Baseline	Our method	Baseline	Our method
1. Moments musicaux	0.18	0.56	0.95	1.00	0.71	0.84
2. Wiegenlied	0.76	1.00	0.83	0.85	0.54	0.69
3. Träumerei	0.60	0.87	0.76	1.00	0.50	0.63
4. An die Freude	0.12	0.73	0.95	1.00	0.22	0.48
5. Barcarolle	0.04	0.54	0.72	0.79	0.24	0.60
	⋮	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮	⋮
Total (100 melodies)	0.46	0.77	0.84	0.90	0.44	0.60

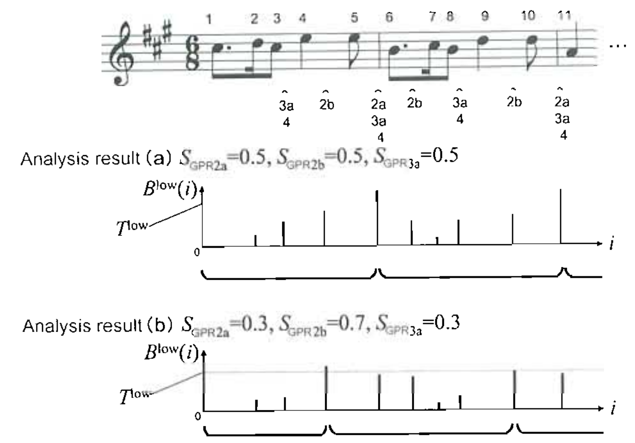

Analysis result (a) S_{GPR2a}=0.5, S_{GPR2b}=0.5, S_{GPR3a}=0.5
Analysis result (b) S_{GPR2a}=0.3, S_{GPR2b}=0.7, S_{GPR3a}=0.3
$B^{low}(i)$
T^{low}
i

Fig. 25. Analysis of Mozart Sonata K. 331.

the parameters are configured so that the $F_{measure}$ of the entire piece will be maximized. Although the second half of the results coincides with the correct data, there are many non-coincident structures in the first half of the piece. At the same time, when we configure the parameters so that the $F_{measure}$ of the first half of the piece will be maximized, there are many non-coincident structures in the second half of the piece (figure 26(b)). This is the only melody investigated where a characteristic of the melody changes in the middle of the piece. It is necessary to further investigate how to analyse such a piece.

Figure 27 shows the analysis results for Bizet, *L'Arlesienne*, *Farandole* as another example of wrong results. In the correct data, there is a group that only has a single event at the end of the piece. However, we implemented exGTTM so that it necessarily holds

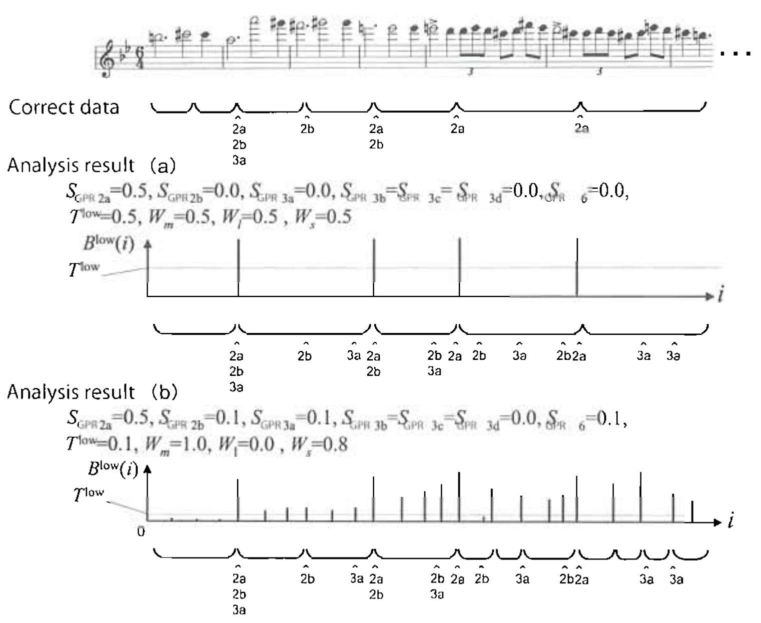

Correct data
Analysis result (a)
S_{GPR2a}=0.5, S_{GPR2b}=0.0, S_{GPR3a}=0.0, S_{GPR3b}=S_{GPR3c}=S_{GPR3d}=0.0, S_{GPR6}=0.0,
T^{low}=0.5, W_m=0.5, W_l=0.5, W_s=0.5
Analysis result (b)
S_{GPR2a}=0.5, S_{GPR2b}=0.1, S_{GPR3a}=0.1, S_{GPR3b}=S_{GPR3c}=S_{GPR3d}=0.0, S_{GPR6}=0.1,
T^{low}=0.1, W_m=1.0, W_l=0.0, W_s=0.8
$B^{low}(i)$
T^{low}
i

Fig. 26. Analysis of Chopin, Ballade op. 23.

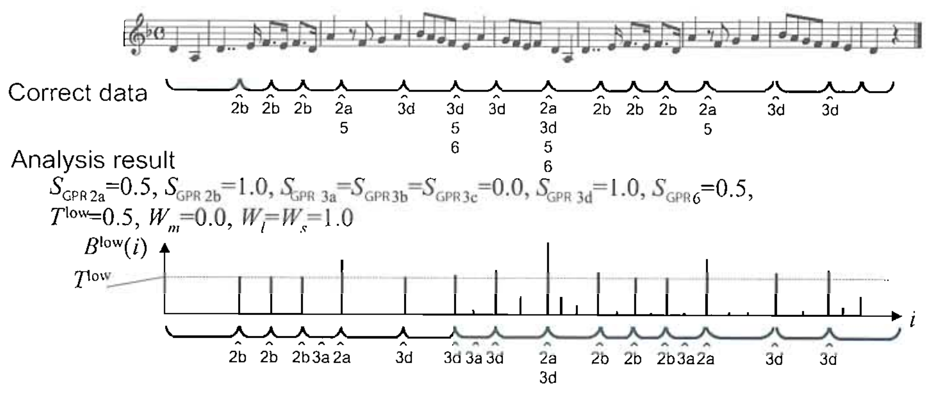

Correct data
Analysis result
S_{GPR2a}=0.5, S_{GPR2b}=1.0, S_{GPR3a}=S_{GPR3b}=S_{GPR3c}=0.0, S_{GPR3d}=1.0, S_{GPR6}=0.5,
T^{low}=0.5, W_m=0.0, W_l=W_s=1.0
$B^{low}(i)$
T^{low}
i

Fig. 27. Analysis of Bizet, *L'Arlesienne*, *Farandole*.

GPR1, which prefers to avoid a single event. It is necessary to further investigate the differences between a piece that has a group containing only a single event and a piece that does not have such a group.

8.4 Acquisition of hierarchical grouping structure

Here, we discuss whether the hierarchical grouping structure was acquired properly. Figures 28 and 29 are the histograms for comparing the correct data and system output by numbers of groups and numbers of grouping hierarchies. The system output tends to have larger numbers of both groups and hierarchies than the correct data, since we implemented exGTTM so that it must hold GPR5, which subdivides groups into two parts. For example, the correct data of Tchaikovsky, *Album pour enfants*, Waltz (Figure 30) contain the three lowest-level groups (notes 1–3, notes 4–6, and notes 7–9) and combine a group at the next grouping level. However, in the system output, notes 1–3 and notes 4–6 are first combined and then notes 1–6 and notes 7–9 are combined. Therefore, the analysis result tends to have larger numbers of both groups and hierarchies than the correct data. In the evaluation data, there are 58 pieces that contain a group consisting of three low-level groups, and the average F_{measure} of the 58 pieces is 0.75, while the average F_{measure} of the other 42 pieces is 0.81. It is necessary to further investigate the algorithm for generating a group that has three low-level groups.

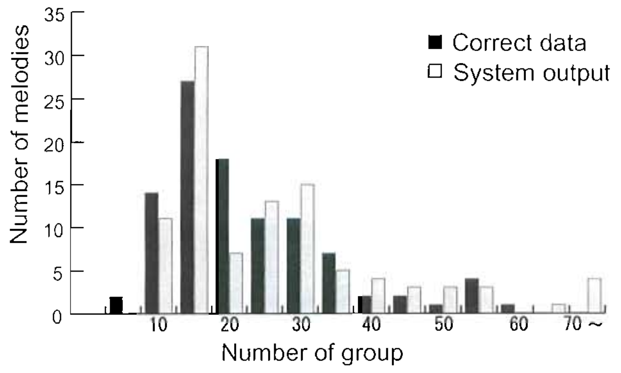


Fig. 28. Numbers of groups in correct data and system output.

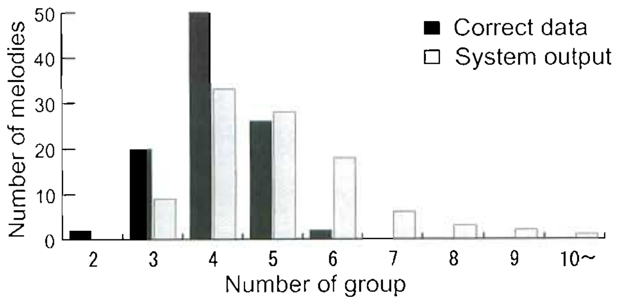


Fig. 29. Numbers of grouping hierarchies in correct data and system output.

8.5 Acquisition of metrical structure

Although the average F_{measure} of the metrical structure analyser is 0.90, the F_{measures} of some pieces are very low. For example, Bach, Toccata and Fugue in D minor (Figure 31) contains a quintuplet, which is not aligned in simple duple/triple time, and cannot be analysed properly in the current exGTTM. In the evaluation data, there are two pieces that contain a quintuplet and one piece that contains septuplets, and the average F_{measure} of the three pieces is 0.25, while the average F_{measure} of the other 97 pieces is 0.92. We plan to extend the system to allow quintuplets and septuplets.

8.6 Acquisition of time-span tree

The average F_{measure} of the time-span tree analyser is 0.60, which is lower than those of the grouping-structure analyser and the metrical-structure analyser. Figure 32 shows the analysis results of Beethoven, Sonata Op. 13, which is an example of a piece having a low F_{measure}. According to the correct data of the time-span tree, we can separate the entire melody into four time-spans, which are notes 1–3, notes 4–9, notes 10–16, and notes 17–23. However, the correct data of the grouping structure also separates the entire melody into four

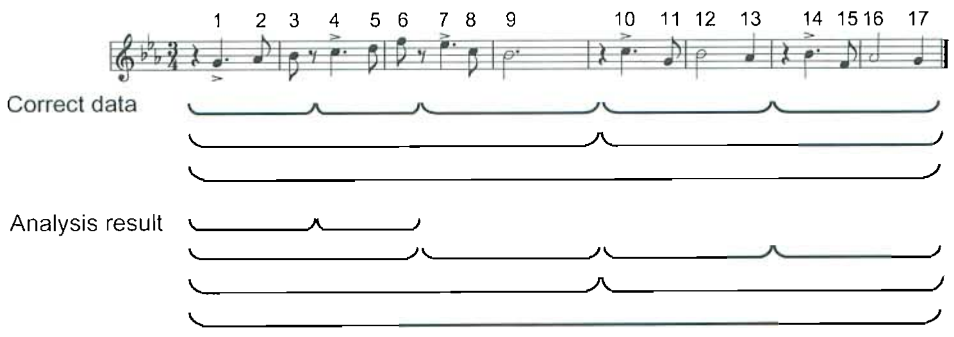


Fig. 30. Analysis of Tchaikovsky, *Album pour enfants*, Waltz.

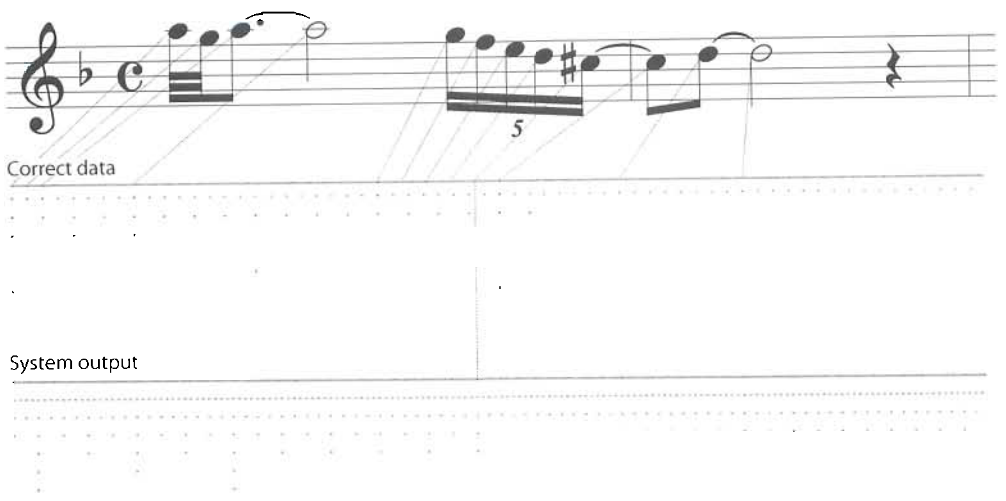


Fig. 31. Analysis of Bach, Toccata and Fugue in D minor.

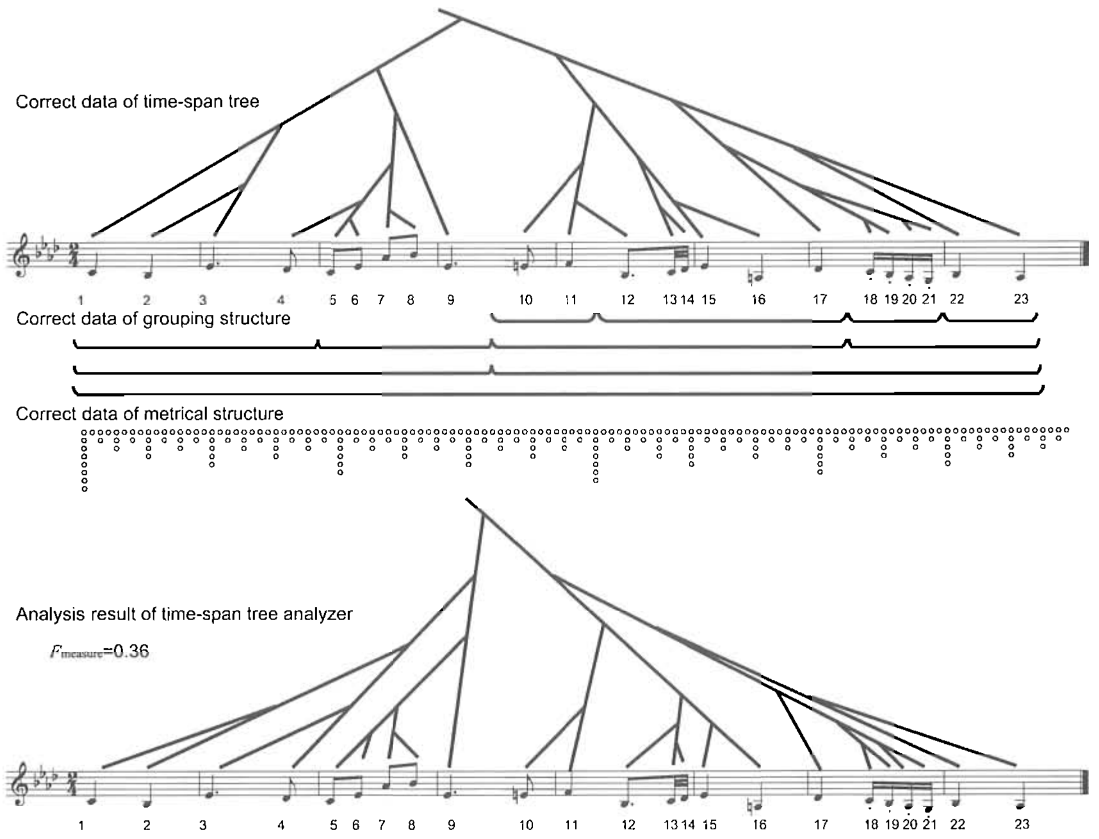

Correct data of time-span tree
Correct data of grouping structure
Correct data of metrical structure
Analysis result of time-span tree analyzer
F_{measure}=0.36
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

Fig. 32. Analysis of Beethoven, Sonata, Op. 13.

groups, which are notes 1–4, notes 5–9, notes 10–17, and notes 18–23. Therefore, these correct data violate time-span segmentation rule 1 (Lerdahl & Jackendoff, 1983, p. 146), which defines that "Every group in a piece is a time-span in the time-span segmentation of the piece." In the current system, we have implemented the time-span segmentation so that time-span segmentation rule 1 will always hold. It is necessary to further investigate the difference between a piece in which time-span segmentation rule 1 holds and a piece in which this rule does not hold.

In the experiment, we cannot acquire the tendency of the parameter set. In other words, we cannot acquire an efficient parameter set for every piece. This result indicates that the parameter set depends on the character of each piece and that we can classify pieces by using the

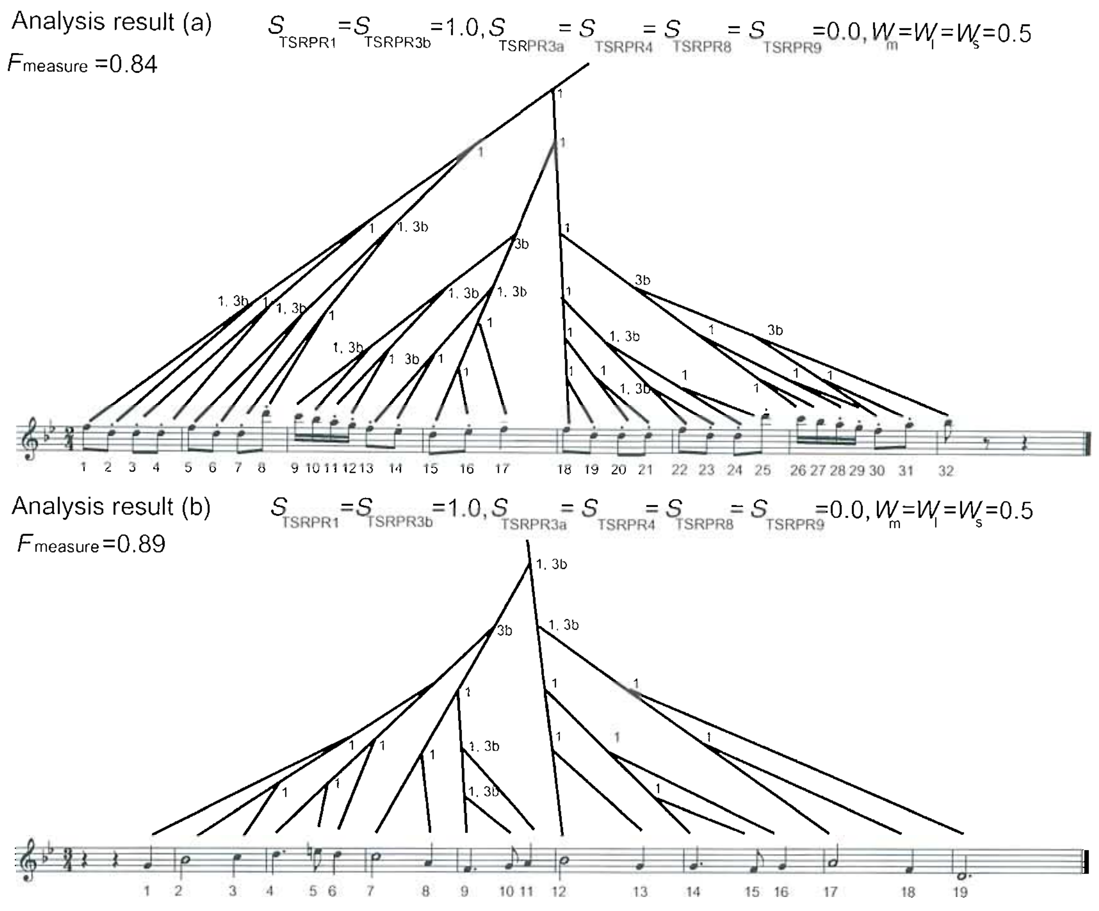

Analysis result (a)
Fmeasure =0.84
S_TSRPR1 =S_TSRPR3b =1.0, S_TSRPR3a = S_TSRPR4 = S_TSRPR8 = S_TSRPR9 =0.0, W_m=W_l=W_s=0.5
Analysis result (b)
Fmeasure=0.89
S_TSRPR1 =S_TSRPR3b =1.0, S_TSRPR3a = S_TSRPR4 = S_TSRPR8 = S_TSRPR9 =0.0, W_m=W_l=W_s=0.5

Fig. 33. Analysis of two pieces having the same parameter sets. (a) Beethoven, Turkish March. (b) English Traditional, Green Sleeves.

parameter set after tuning. Figure 33 shows the analysis results of two pieces, which are tuned with the same parameters, i.e. Beethoven, Turkish March and English Traditional, Green Sleeves. The numbers at the separations of the tree in Figure 33 indicate the rules that hold. As a result of comparing two pieces, the same rules hold, i.e. TSRPR1 and TSRPR3b. We plan to investigate the classifications of genre and composer by using the parameter sets.

9. Conclusion

We developed a music analysing system called ATTA that derives the time-span tree of GTTM. The following three points are the main results of this study.

- **Extended GTTM proposed and implemented**
  We proposed an extended GTTM for computer implementation. The difficulty with the computer implementation of GTTM has been clarified (Heikki, 2000), but no radical solution has been proposed. We re-formalized the rules using a numerical expression with adjustable parameters so that it can separate the definition and ambiguity from the analysed material.

  We implemented an actual working system to acquire the hierarchical grouping structure, metrical structure, and time-span tree of music, based on GTTM. This system, called ATTA, automatically acquires the time-span tree by configuring the parameters without manually analysing them by experts in musicology. ATTA is the first application for automatically acquiring a time-span tree. We can benchmark ATTA against other systems, since the ATTA can be used through the internet by using a CGI application, as described in Section 7.2.
- **Full externalization and parameterization**
  As we discussed in Section 3.3.1, we have distinguished three kinds of parameters. Among them, we have fully externalized the parameters of the first and the second categories, i.e. those which are explicitly or implicitly mentioned in GTTM (Lerdahl & Jackendoff, 1983). Also, we could supplement those of the third category sufficiently to reify the preference rules. We would like to justify this claim as follows.

  Our working hypothesis is that we should apply all of the rules to the applicable parts. Because some rules refer to local structures and others to global ones, the process necessarily becomes from both the top-down and bottom-up directions, and it becomes entangled in spaghetti of dependency as shown in Figures 6, 14 and 19. Thus, our objective becomes unravelling all of these messy relations, to make the procedure deterministic. Toward this goal, we first gave a threshold (the first category) to each rule to judge whether the rule is applicable. Then, we gave a weight (the second category) to each rule to fix the order of rule application. Lastly, the global constraints represented by these parameters were recalculated every time the processing for generating a hierarchical structure moved onto an adjacent layer. As far as the process was made to be deterministic, we contend that those explicit/implicit parameters were fully externalized.

The parameters of the third category mainly concern practical implementation and are rather independent of GTTM; thus, the parameters are insignificant from the musicological point of view. Since there remains no ambiguous step in the analysing process, we consider that they are sufficiently elucidated.

- **A set of correct data constructed and evaluated**
We assembled a set of one hundred correct data items, which is the greatest database of analysed results from GTTM to date. We plan to make this database publicly available in the near future.

Our experimental results show that, as a result of configuring the parameters, our music analyser outperforms the baseline F_{measure}. The set of parameters tuned for a certain family of music pieces would likely reflect the common features of that family. Thus, the idealized parameter set for a music family, if any, would presumably analyse a new piece correctly, prior to human analysis.

Among the four sub-theories of GTTM, i.e. grouping-structure analysis, metrical-structure analysis, time-span reduction, and prolongational reduction, we have implemented the first three sub-theories in this study. Since the prolongational reduction involves harmonic stability, it requires the analysis of tonal structure, including chord and/or key recognition and its cadential progression. Because our objective in this study is to clarify and exemplify the computability of GTTM by externalization and parameterization, we have avoided the complication of tonal analysis; namely, we have postponed the mechanization of the fourth theory and dared to stay in the analysis of monophonic music. As a result, in reductions heads are restricted to the ordinary ones, and *fusion*, *transformation*, and *cadential retention* (Lerdahl & Jackendoff, 1983, pp. 152–158) have been disregarded. Naturally, we are aware that the result of prolongational reduction would be fed back to the time-span tree. Therefore, the implementation of the fourth theory is an urgent and emergent goal of our future research.

We plan to develop further systems, using time-span trees and the results of the music analyser, for other musical tasks, such as searching, harmonizing, voicing, and ad-libbing. Such systems will help to evaluate the effectiveness of implementing GTTM as a way to provide musical knowledge.

References

Cambouropoulos, E. (2001). The Local Boundary Detection Model (LBDM) and its application in the study of expressive timing. In *Proceedings of ICMC2001*, Havana, pp. 290–293.

Cooper, G. & Meyer, L.B. (1960). *The rhythmic structure of music*. Chicago, IL: The University of Chicago Press.

Cope, D. (1996). *Experiments in musical intelligence*. Madison, WI: A-R Editions, Inc.

Ferrand, M., Nelson, P. & Wiggins, G. (2003). Memory and melodic density: a model for melody segmentation. *Proceedings of XIV CIM 2003*, Firenze, pp. 95–98.

Goto, M. (2001). An audio-based real-time beat tracking system for music with or without drum-sounds. *Journal of New Music Research*, *30*(2), 159–171.

Hamanaka, M. & Hirata, K. (2002). Applying Voronoi diagrams in the automatic grouping of polyphony. *Information Technology Letters*, *1*(1), 101–102.

Hamanaka, M., Hirata, K. & Tojo, S. (2004). Automatic generation of grouping structure based on the GTTM. *Proceedings of ICMC2004*, Miami, pp. 141–144.

Hamanaka, M., Hirata, K. & Tojo, S. (2005a). Automatic generation of metrical structure based on the GTTM. *Proceedings of ICMC2005*, Barcelona, pp. 53–56.

Hamanaka, M., Hirata, K. & Tojo, S. (2005b). ATTA: automatic time-span tree analyzer based on extended GTTM. *Proceedings of ISMIR2005*, London, pp. 358–365.

Heikki, V. (2000). *Lerdahl and Jackendoff revisited*. Available online at: http://www.cc.jyu.fi/~heivalko/articles/lehr_jack.htm

Hewlett, W.B. (Ed.) (1998). Melodic similarity: concepts, procedures, and application. *Computing in musicology, volume 11*, Cambridge, MA: The MIT Press.

Hirata, K. & Aoyagi, T. (2003). Computational music representation on the generative theory of tonal music and the deductive object-oriented database. *Computer Music Journal*, *27*(3), 73–89.

Hirata, K. & Hiraga, R. (2003). Ha-Hi-Hun plays Chopin's Etude. *Working Notes of IJCAI-03 Workshop on Methods for Automatic Music Performance and their Applications in a Public Rendering Contest*, pp. 72–73.

Hirata, K. & Matsuda, S. (2003). Interactive music summarization based on generative theory of tonal music. *Journal of New Music Research*, *32*(2), pp. 165–177.

Lerdahl, F. & Jackendoff, R. (1983). *A generative theory of tonal music*. Cambridge, MA: The MIT Press.

Marsden, A. (2005). Generative structural representation of tonal music. *Journal of New Music Research*, *34*(4), pp. 409–428.

Narmour, E. (1990). *The analysis and cognition of basic melodic structure*. Chicago, IL: The University of Chicago Press.

Nord, T. (1992). Toward theoretical verification: developing a computer model of Lerdahl and Jackendoff's generative theory of tonal music. PhD thesis, The University of Wisconsin, Madison, USA.

PG Music Inc. (2007). Finale. Available online at: http://www.pgmusic.com/finale.htm

Recordare, LLC. (2007a). MusicXML 1.1 Tutorial. Available online at: http://www.recordare.com/xml/musicxml-tutorial.pdf

Recordare, LLC. (2007b). Dolet 3 for Finale. Available online at: http://www.recordare.com/finale/index.html

Rosenthal, D. (1992). Emulation of human rhythm perception. *Computer Music Journal*, *16*(1), 64–76.

Russell, S. & Novig, P. (2002). *Artificial intelligence: a modern approach*. Englewood Cliffs, NJ: Prentice Hall.

Schenker, H. (1979). In E. Oster (Trans. and Ed.) *Free composition (Der freie Satz)*. New York: Longman.

Selfridge-Field, E. (1998). Conceptual and representational issues in melodic comparison. *Computing in Musicology 11* (pp. 3–64). Cambridge, MA: The MIT Press.

Stammen, D.R. & Pennycook, B. (1994). Real-time segmentation of music using an adaptation of Lerdahl and Jackendoff's grouping principles. *Proceedings of ICMPC1994*, Liege, pp. 269–270.

Temperley, D. (2001). *The cognition of basic musical structures*. Cambridge, MA: The MIT Press.

Todd, N. (1985). A model of expressive timing in tonal music. *Musical Perception*, *3*(1), 33–58.

W3C (2001). XML Linking Language (XLink) Version 1.0. Available online at: http://www.w3.org/TR/xlink/

W3C (2002). XML Pointer Language (XPointer). Available online at: http://www.w3.org/TR/xptr/

Widmer, G. (1993). Understanding and learning musical expression. *Proceedings of ICMC1993*, Tokyo, pp. 268–275.