σGTTM III: Learning based Time-span Tree Generator based on PCFG

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

[1]Kyoto University,
hamanaka@kuhp.kyoto-u.ac.jp,
[2]Future University Hakodate,
hirata@fun.ac.jp,
[3]JAIST,
tojo@jaist.ac.jp,
http://gttm.jp/

Abstract. We propose an automatic analyzer for acquiring a time-span tree based on the generative theory of tonal music (GTTM). Although analyzer based on GTTM was previously proposed, it requires manually tweaking the 46 adjustable parameters on a computer screen in order to analyze them properly. We reformalized the time-span reduction in GTTM based on a statistical model called probabilistic context-free grammar, which enables us to acquire the most probabilistic time-span tree. We applied leave-one-out cross validation using three hundred sets training data, which revealed that our analyzer outperformed the previous one.

Keywords: A generative theory of tonal music (GTTM), probabilistic context-free grammar (PCFG), time-span tree.

1 Introduction

This paper describes a method for automatic generation of a time-span tree based on the generative theory of tonal music (GTTM) [1]. The main advantage of our method is that it is based on probabilistic context-free grammar (PCFG) [2], and we can therefore acquire a model that enables us to generate a time-span tree by statistically learning training data that has been analyzed manually by a musicologist.

Generally, a piece of music will have more than one interpretation, and dealing with such ambiguity is a major obstacle when implementing a music theory on a computer. A statistical model is suitable for constructing a music analyzer that allows such ambiguity. In other words, when we introduce a statistical model in a musical analyzer, we can compare the likelihood of one interpretation to other interpretations. One such probabilistic model we introduce here is PCFG, which is used for syntactic analysis of natural language.

PCFG consists of multiple rules, and each rule has a probability. The probability of a sentence is derived from multiplications of probabilities of applied rules for generat-ing the sentence. Finding a rule set that can generate the sentence is called "parsing," and a tree structure that indicates the parse result is called a "parse tree."

The time-span tree is a result of analyzing GTTM, which is a binary tree in each leaf connecting a note lined in time order. Each middle node between roots and leaves has labels that indicate which note is salient between two notes that connect directly to the node. Therefore, a time-span tree can acquire a reduction melody by omitting non-salient notes.

In this study, we regard the time-span tree as a parse tree of a melody. We can generate a melody by using PCFG. Then, the reduction process is an inverse problem of the generation. We first set 645 PCFG rules and learned the generation probabilities of the rules by using training data that had been analyzed manually by a musicologist. Then we calculated the total probability of each parse tree of all the well-formed time-span trees and selected the maximum one as the most appropriate time-span tree to solve the inverse problem.

We applied leave-one-out cross validation using 300 sets of training data. The results indicated an average accuracy of 0.76, which outperformed the previous GTTM analyzer.

2 Related work

We briefly look back at the history of cognitive music theory. The implication-realization model (I-R model) proposed by Eugene Narmour abstracts and expresses music according to symbol sequences from information from a musical score [3, 4]. Recently, the IRM has been implemented on computer and can be used to acquire the chain structures of I-R model from a score [5]. In contrast, Schenkerian analysis is used to analyze the deeper structures called " Urlinie " (fundamental line) and " Ursatz " (fundamental structure) from the music surface [6]. Short segments of music can be analyzed using Schenkerian analysis on a computer [7]. Another approach constructs a music theory for computer implementation [8, 9].

The main advantage of analysis by GTTM is that it can acquire the tree structures called time-span and prolongation trees. The time-span or prolongation tree provides a summarization of a piece of music, which can be used as the representation of an abstraction, resulting in a music retrieval system [10]. It can also be used for performance rendering [11] and reproducing music [12]. Additionally, the time-span tree can be used for melody prediction [13] and melody morphing [14].

Some other studies have applied PCFG to analyze music, for example, chord analysis for jazz [15], analysis of the metrical structure [16], and automatic transcription [17]. These studies [Granroth 2012,Tanji 2008,Kameoka 2012] show the usefulness of using PCFG in musical analysis. Actually, the GTTM [1] has a concept of generation that is also expressed in the title. However, no one has yet regarded a time-span tree as a parse tree or estimated the most probabilistic time-span tree.

We constructed four types of GTTM analyzers: ATTA, FATTA, σGTTM, and σGTTMII. We extended the original theory of GTTM with a full externalization and parameterization and proposed a machine-executable extension of the GTTM called exGTTM [18]. The externalization includes introducing an algorithm to generate a hierarchical structure of the time-span tree in a mixed top-down and bottom-up manner, and the parameterization includes introducing a parameter for controlling the priorities of rules in order to avoid conflict among the rules, as well as parameters for controlling the shape of the hierarchical time-span tree. We implemented exGTTM on a computer called the ATTA (automatic time-span tree analyzer), which can output multiple analysis results by configuring the parameters.

Although the ATTA has adjustable parameters for controlling the weight or priority of each rule, these parameters have to be set manually. This takes a long time because finding the optimal values of the settings themselves takes a long time. We discuss the problem of ATTA in detail in 3.3.

The FATTA (full-automatic time-span tree analyzer) can automatically estimate the optimal parameters by introducing a feedback loop from higher-level structures to lower-level structures on the basis of the stability of the time-span tree [19]. The FATTA can output only one analysis result without manual configuration. However, the FATTA performance is not good enough for analyzing time-span trees.

We also developed σGTTM, a system that can detect the local grouping boundaries in GTTM analysis, by combining GTTM with statistical learning [20]. The σGTTM system statistically learns the priority of GTTM rules from 100 sets of score and grouping structure data analyzed by a musicologist; it does this by using a decision tree. Its performance, however, is insufficient because it can construct only one decision tree from 100 data sets and cannot output multiple results.

The σGTTM II system assumes that a piece of music has multiple interpretations, and thus, it constructs multiple decision trees (each corresponding to an interpretation) by iteratively clustering the training data and training the decision trees. The performance of the σGTTM II system outperformed both the ATTA and σGTTM systems [21]. However, σGTTM and σGTTM II are only suitable for grouping structures and cannot acquire time-span trees.

3 Time-span reduction and its implementation problem

We use the grouping and metrical structures of music to derive a time-span tree. The grouping structure is intended to formalize the intuitive belief that tonal music is organized into groups that are in turn composed of subgroups. These groups are graphically presented as several levels of arcs below a music staff. The metrical structure describes the rhythmic hierarchy of the piece by identifying the position of strong beats at different levels such as those of a quarter note, half note, a measure, two measures, and four measures. Strong beats are illustrated as several levels of "dots" below the musical staff. The time-span tree is a binary tree, which is a hierarchical structure describing the relative structural importance of notes that differentiate the essential parts of the melody from the ornamentation (Fig. 1). For example, the left side of Fig. 2 depicts a simple melody and its tree. The time-span (designated as ¡—¿) is represented by a single note, called a head, which is designated here as "C4." In the tree, the essential notes are connected to a branch nearer to the root of the tree. In contrast, the ornamentation notes are connected to the leaves of the tree. In a separation, we hereafter call the branch "primary" and the leaf "secondary" (Fig. 3).

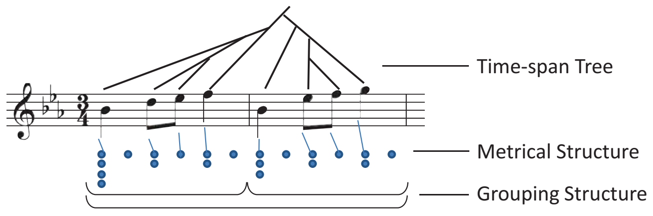


Fig. 1. Time-span tree, metrical structure, and grouping structure.

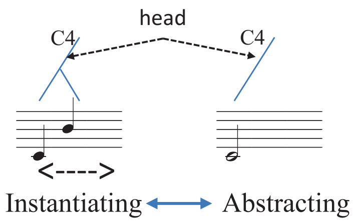


Fig. 2. Subsumption relation of melodies.

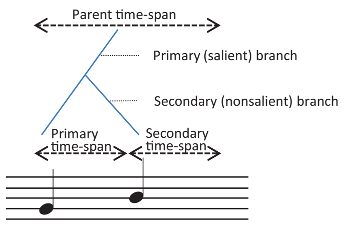


Fig. 3. Primary and secondary time-span trees.

3.1 Time-span segmentation

Before the time-span reduction, the time-span segmentation divides the entire piece into hierarchical time-spans. We show the division procedure in Fig. 4, which involves the following steps:

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

In [1], there are two rules for time-span segmentation, which are called segmentation rule 1 and segmentation rule 2. The former corresponds to the first item and the latter to the second item.

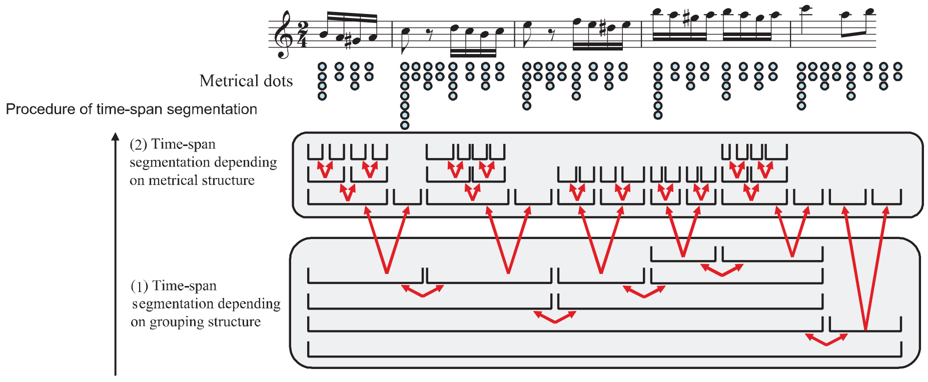

Metrical dots
Procedure of time-span segmentation
(2) Time-span segmentation depending on metrical structure
(1) Time-span segmentation depending on grouping structure

Fig. 4. Time-span segmentation.

3.2 Time-span reduction

As a result, a music piece is formed into a binary 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.

Nine rules are defined for the time-span reduction preference rules that indicate superiority of one tree over another. These rules consist of local rules and broad rules.

For example, both TSRPR1 and TSRPR5 are rules related to the metrical structure. However, TSRPR1 is a local rule, and TSRPR5 is a more global rule.

TSRPR1 (Metrical Position) Of the possible choice for head of a time-span tree T, prefer a choice that is in a relatively strong metrical position.

TSRPR5 (Metrical Stability) In choosing the head of a time-span T, prefer a choice that results in a more stable choice of metrical structure.

The biggest problem when implementing time-span reduction on computer is that there is little information on how to combine local and broad rules and how to construct hierarchical time span trees.

3.3 Problems of time-span reduction in ATTA

In the ATTA we introduced adjustable parameters for control the strength of each rule of time-span preference rule in order to overcome the problem in 3.2. A hierarchical time-span tree is constructed by iterating the calculation of the plausibility of the head for the current heads and choosing the next level heads (Fig. 5).

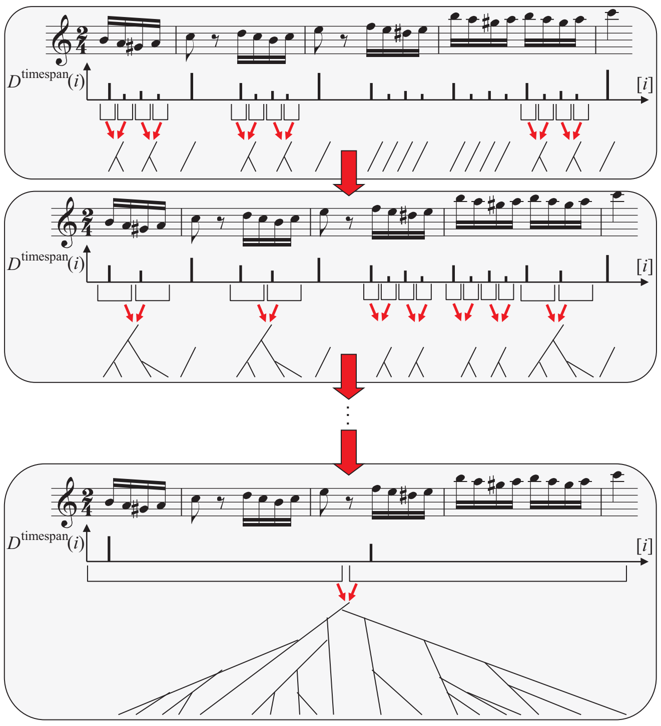


Fig. 5. Selecting the next-level heads in the time-span reduction.

However, the performance of time-span reduction in ATTA was not sufficient. We investigated some pieces of music for which the time-span reduction performance was not very good, and the results indicated that many of them had a weak beat becomes a head (primary) near the leaves of the tree and a strong beat becomes a head near the root of the tree (Fig. 6). In other words, the ATTA cannot perform well in songs where the important rules are different depending on the branching which is near the leaves or the root of the tree.

We can consider two ways of solving this problem. The first one is to introduce a more adjustable parameter that enables us to control the strength of rules at each level of the time-span. However, this idea is not practical because it is difficult to adjust each parameter manually. The second way is described in the next section.

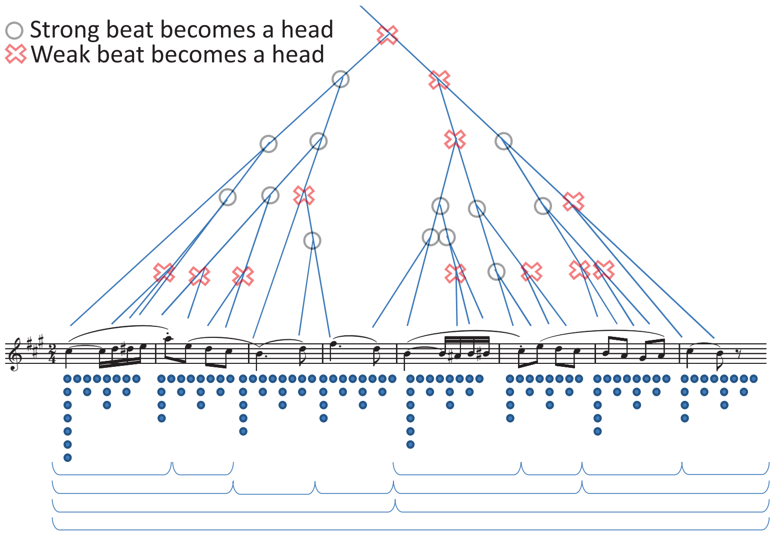


Fig. 6. Example of a piece that is not good performance.

4 σGTTMIII: Learning based time-span tree generator

The time-span tree can extract an abstracted melody by reducing ornamentation notes. An example of time-span reduction is shown in Fig. 7. The time-span tree in the figure is from melody A, which embodies the results of GTTM analyses. We can obtain an abstracted melody B by slicing the tree in the middle and omitting notes that are connected to branches under line B. In the same manner, if we slice the tree higher up at line C, we can get a more abstracted melody C.

If we apply this reduction process in the inverse direction, we can use it as a generation process as follows (Fig. 8).

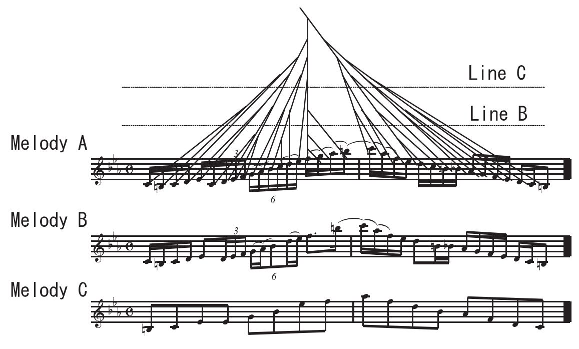


Fig. 7. Time-span reduction.

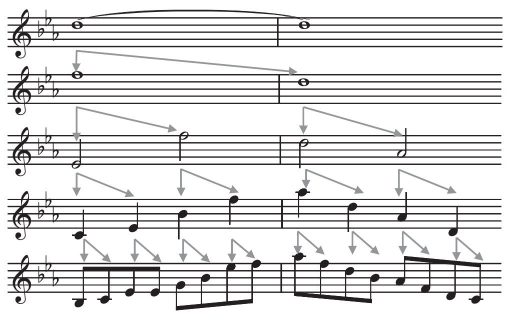


Fig. 8. Generation process of note sequence.

1. Identify the note in which the time-span (length) is the same as the whole piece of music.
2. Separate the time-span of the note and make a primary note and a secondary note.
3. Repeat 2 recursively to the leaves of the tree.

By expressing the above generation process using a probabilistic model, we can acquire the most probabilistic time-span tree.

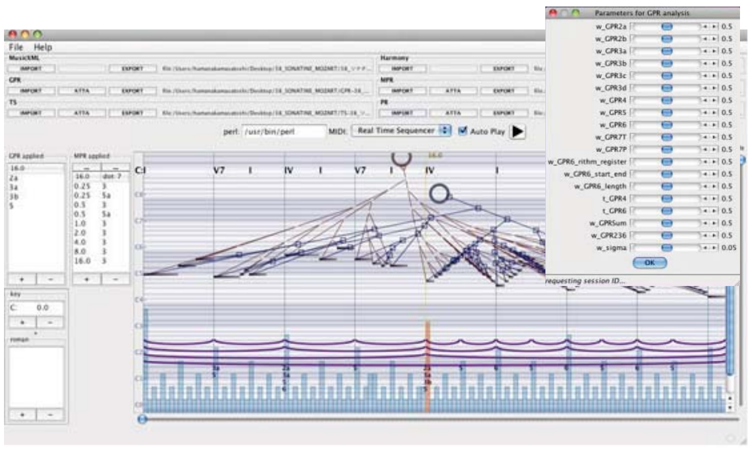
4.1 Training data

The training data we used for the probabilistic model was a musical structural database based on GTTM that we constructed [1].

We collected 300 8-bar-long monophonic classical music pieces that included notes, rests, slurs, accents, and articulations entered manually using music notation software called "Finale" [23]. We exported the MusicXML by using a plugin called "Dolet." The 300 whole pieces and the 8 bars were selected by a musicologist.

We asked a musicology expert to manually analyze the score data faithfully with regard to the GTTM by using the manual editor in the GTTM analysis tool (Fig. 9) in order to assist in editing the grouping structure, metrical structure, and time-span tree. Three other experts crosschecked these manually produced results.

The analyzer and database can be downloaded at `http://www.gttm.jp/`



Parameters for GPR analysis

w_GPR2a	0.5
w_GPR2b	0.5
w_GPR3a	0.5
w_GPR3b	0.5
w_GPR3c	0.5
w_GPR3d	0.5
w_GPR4	0.5
w_GPR5	0.5
w_GPR6	0.5
w_GPR7T	0.5
w_GPR7P	0.5
w_GPR6_rithm_register	0.5
w_GPR6_start_end	0.5
w_GPR6_length	0.5
t_GPR4	0.5
t_GPR6	0.5
w_GPRSum	0.5
w_GPR236	0.5
w_sigma	0.05

Fig. 9. Interactive GTTM analyzer.

4.2 PCFG model for generating time-span tree

We introduce here a probabilistic context-free grammar (PCFG), which we used to construct a probabilistic generation model of a melody. The PCFG has multiple generation rules with probabilities that can represent the separation of primary and secondary notes in each node of a time-span tree.

The PCFG of G can be defined by a quintuple:

$$G = \{T, M, S, R, P\} \tag{1}$$

where T is the set of terminal symbols, M is the set of non-terminal symbols, S is the start symbol, R is the set of production rules, and P is the set of probabilities on generation rules.

***T*: terminal symbols.** Notes are the terminal symbols.
***M*: non-terminal symbols.** Time-spans are the non-terminal symbols.
***S*: start symbol.** The start symbol of PCFG for generating a time-span tree is the length of an entire piece of music without any rests, which means the length of the time-span of the whole piece.

***R*: production rules.** There are two kinds of production rules. The first kind involves a time-span from which two time-spans are generated. We call this rule the time-span separation rule. The sum of the length of two time-spans is the same as the length of the original time-span. There are several ways to generate two time-spans from one time-span. The second kind of rule involves a time-span to generate a note, which we call the note generation rule. The length of the original time-span and the length of the generated note are the same (Fig. 10).

***P*: probabilities of generation rules.** Each production rule has a probability. For example, the probability of generating a time-span that has the length of 32nd generate 32nd note is almost 1.00. On the other hand, the probability of generating a time span that has the length of a double note generate double note is almost 0.00 because there are many other production rules for a double note (Fig. 10).

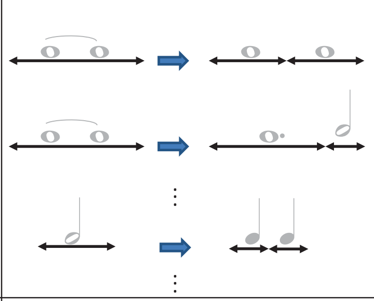
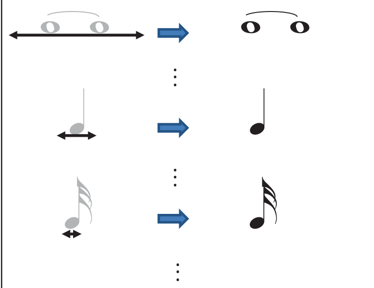
	Production Rules	Probability
Time-span Separation rules		0.35 0.12 ⋮ 0.62 ⋮
Note generation rules		0.01 ⋮ 0.44 ⋮ 0.99 ⋮

Fig. 10. Example of production rules and its probabilities.

4.3 Design rules of PCFG for generating time-span tree

We have so far only discussed the length of a note in the basic PCFG model described in 4.2. Other relevant information is as follows.

Primary and secondary. When two time-spans are generated according to the generation rules, one is assigned as a primary time-span and the other is assigned as a secondary time-span.

Pitch. When the primary and secondary time-spans are generated, the primary pitch will inherited and second pitch will generated.

Order of time-spans. There are two orders when primary and secondary time-spans are generated. In one, the primary time-span is generated before the secondary time-span. In the other, the secondary time-span is generated before the primary time-span.

Numbers of dots. As described in 3.2, the metrical structure strongly affects the generation of the time-span. Therefore, the numbers of dots in the primary and secondary time-spans should be included in the model.

Use of the above information in the naive implementation of the rules of PCFG results in too many rules, and the probability of most of them is zero because we have limited numbers of training datasets. We solve this problem of space training data by abstracting the rules as follows.

3 types of pitch change. We graded the changes in pitch of primary and secondary time-spans as up, down, or the same.

7 types of duration ratio. We graded the length ratio of primary and secondary time-spans as one closest to the 4 times, 3 times, 2 times, 1 time, 1/2 time, 1/3 time, and 1/4 time.

2 orders of time-spans. One order is that the primary time-span is before the secondary; the other is that the secondary time-span is before the primary.

3 types of numbers of dots. We graded the dots of primary and secondary time-spans in three groups: primary has many dots, secondary has many dots, and primary and secondary have the same number of dots.

5 types of head time-span length. We graded the length of the time-span before separating it into primary and secondary as the one closest to sixteenth, eighth, quarter, half, whole, and double time.

We established 645 PCFG rules in total, which consist of 630 (=3x7x2x3x5) time-span separation rules and 15 (=3x5) note generation rules.

4.4 Generation of time-span tree by using PCFG

The probability of each PCFG rule was achieved using supervised learning by counting 19,296 nodes of 300 time-span trees in our database. Then we were able to obtain the most probabilistic time-span tree by calculating the total probability of each parse tree of all the well-formed time-span trees and selecting the maximum one (Fig. 11).

Generating all the well-formed time-span trees requires substantial computing time because the combinations of trees increase exponentially when the numbers of notes increase. To reduce the computing time, we parallelized eight processes for generating time-span trees for each piece of music. By using the PCFG, we needed three weeks to acquire plausible time-span trees of 100 pieces of music in the GTTM database using a PC cluster (16 machines of Intel Xeon E5-2430@2.00 GHz 12core). The computing time depended on the musical pieces; the longest was two weeks, and the shortest one was two minutes.

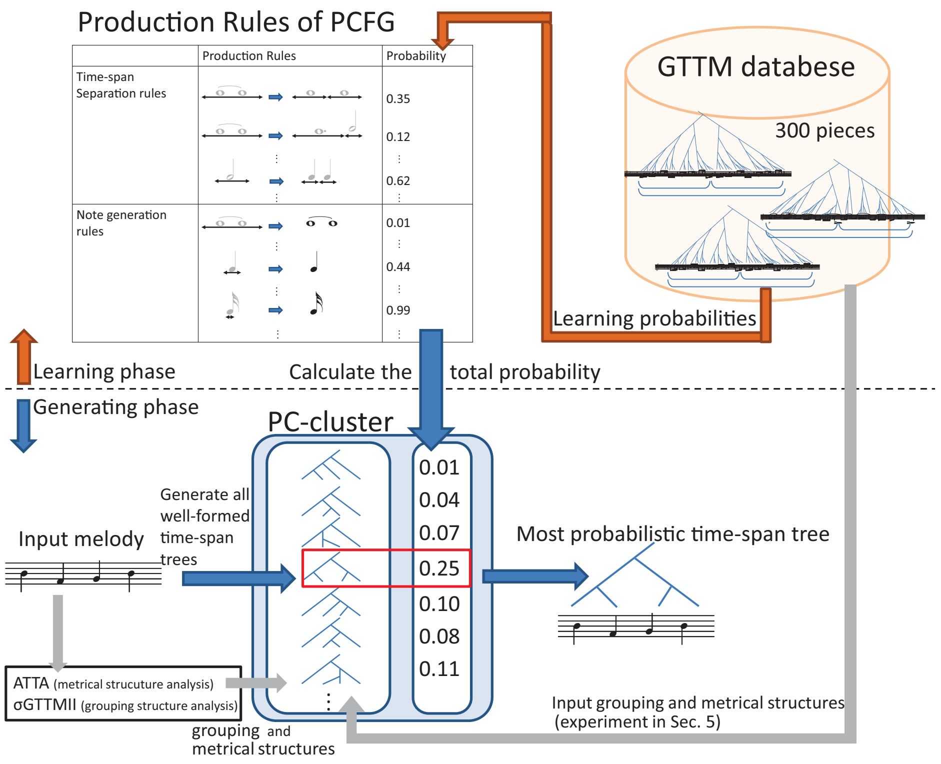

Production Rules of PCFG
Production Rules
Probability
Time-span Separation rules
0.35
0.12
0.62
Note generation rules
0.01
0.44
0.99
GTTM databese
300 pieces
Learning probabilities
Learning phase
Calculate the total probability
Generating phase
PC-cluster
Generate all well-formed time-span trees
Input melody
0.01
0.04
0.07
0.25
0.10
0.08
0.11
Most probabilistic time-span tree
ATTA (metrical strucuture analysis)
σGTTMII (grouping structure analysis)
grouping and metrical structures
Input grouping and metrical structures (experiment in Sec. 5)

Fig. 11. Overview of σ GTTMIII.

5 Experimental results

We evaluated the performance of σGTTMIII by leave-one-out cross validation in order to compare its accuracy with that of ATTA, a previously developed time-span tree analyzer [18]. We used the same 100 pieces of music as used to evaluate ATTA, and they were numbered from 1 to 100 in the 300-piece database. When calculating accuracy, we did not consider the possibility that a low-level error would propagate up to a higher level; we counted wrong answers without regard to the differences in time-span levels.

$$Accuracy = \frac{\text{Numbers of matched node in the time-span tree}}{\text{Numbers of node in the time-span tree}} \quad (2)$$

The results of our experiments are given in Table 1. The ATTA has two kinds of results because the accuracy of ATTA varies depending on its adjustable parameters.

It took us an average of about 10 min per piece to find the plausible tuning for the set of parameter. The σGTTMIII outperformed ATTA in average accuracy in both the baseline and after tuning the parameters. The σGTTMIII outperformed the baseline performance of ATTA for all the pieces. After the parameters were tuned, the ATTA outperformed σGTTMIII in a few of the pieces.

Table 1. Accuracies of ATTA and σ GTTMIII.

Melodies	Baseline performance of ATTA	ATTA with configured parameters	σGTTMIII
1. Moments Musicaux	0.71	0.84	0.88
2. Wiegenlied	0.54	0.69	0.78
3. Traumerei	0.50	0.63	0.84
4. Sinfonie Nr.9 d moll Op.125 4.Satz An die Freude	0.22	0.48	0.68
5. The Nutcracker Suite Op.71a No.8 Waltz of the Flowers	0.42	0.91	0.72
	⋮	⋮	⋮
Total (100 melodies)	0.44	0.60	0.76

6 Conclusion

We described σGTTMIII, a time-span tree generator based on PCFG. We set 645 rules for generating time-spans and after carrying out supervised learning of the rules of PCFG, the σGTTMIII outperformed ATTA, the previous time-span tree analyzer, in average accuracy.

Some applications such as melody morphing or melody summarization require the use of a time-span tree [10–13]. However, such applications are not practical because the accuracy of the previous time-span analyzer is not sufficient and requires manual parameter tuning.

We plan to construct an application for automatic melody morphing or summarization. We also plan to achieve un-supervised learning of rules to improve the accuracy.

Acknowledgments. This work was supported in part by JSPS KAKENHI Grant Numbers 23500145, 25330434, and 25700036 and PRESTO, JST.

References

1. Lerdahl, F., and Jackendoff, R.: A Generative Theory of Tonal Music. MIT Press, Cambridge (1983)
2. Charniak, E.: Tree-bank grammars. In: Proceeding of Association for the Advancement of Artificial Intelligence (AAAI-96), pp. 1032–1036, Portland (1996)
3. Narmour E: The Analysis and Cognition of Basic Melodic Structure. University of ChicagoPress (1990).
4. Narmour E: The Analysis and Cognition of Melodic Complexity. The University of ChicagoPress (1992)
5. Yazawa, S., Terasawa, H., Hamanaka, M., Hirata, K., and Tojo, S.: Analysis the chain structures of Implication-Realization Model in a melody, IPSJ SIG Technical Report, 2010-MUS-87, No. 1 (2002) (in Japanese).
6. Schenker, H. Der frei Satz. Vienna: Universal Edition, 1935. Published in English as Free Composition, translated and edited by E. Oster, New York: Longman (1979)
7. Alan, M.: Software for Schenkerian Analysis, In: Proceeding of the 2011 International Computer Music Conference (ICMC2011), pp. 673–67 (2011)
8. Temperley, D.: The Congnition of Basic Musical Structures. MIT Press, Cambridge (2004)
9. Lerdahl, F.: Tonal Pitch Space, Oxford University Press (2001)
10. Hirata, K., and Matsuda, S.: Interactive Music Summarization based on Generative Theory of Tonal Music. Journal of New Music Research, 32:2, 165–177 (2003)
11. Hirata, K., and Hiraga, R.: 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 (2003)
12. Hirata, K., and Matsuda, S.: Annotated Music for Retrieval, Reproduction, and Sharing, In: Proceeding of International Computer Music Conference (ICMC2004), pp. 584–587 (2004)
13. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Expectation Method based on GTTM and TPS, In: Proceeding of the 2008 International Society for Music Information Retrieval Conference (ISMIR2008), pp. 107–112 (2008)
14. Hamanaka, M., Hirata, K., and Tojo, S.: Melody Morphing Method based on GTTM, In: Proceeding of the 2008 International Computer Music Conference (ICMC2008), pp. 155–158 (2008)
15. Granroth, W,M., and Steedman, M.: Statistical Parsing for Harmonic Analysis of Jazz Chord Sequences, In: Proceeding of the 2012 International Computer Music Conference (ICMC2012), pp. 478–485 (2012)
16. Tanji, M., Ando, D., and Iba, H.: Improving Metrical Grammar with Grammar Expansion, In: Australasian Conference on Artificial Intelligence, AI08, LNAI 5360, Springer, pp. 180–191 (2008)
17. Kameoka, H., Ochiai, K., Nakano, M., Tsuchiya M., Sagayama, S.: Context-free 2d Structure Model of Musical Notes for Bayesian Modeling of Polyphonic Spectrograms, In: Proceeding of the 2012 International Society for Music Information Retrieval Conference (ISMIR2012), pp. 307–312 (2012)
18. Hamanaka, M., Hirata, K., and Tojo, S.: Implementing 'A Generative Theory of Tonal Music', Journal of New Music Research, 35:4, 249–277 (2006)

19. Hamanaka, M., Hirata, K., and Tojo, S.: FATTA: Full Automatic Time-span Tree Analyzer, In: Proceedings of the 2007 International Computer Music Conference (ICMC2007), pp. 153–156 (2007)
20. Miura, Y., Hamanaka, M., Hirata, K., and Tojo, S.: Use of Decision Tree to Detect GTTM Group Boundaries, In: Proceedings of the 2009 International Computer Music Conference (ICMC2009), pp. 125–128 (2009)
21. Kanamori, K., and Hamanaka, M.: Method to Detect GTTM Local Grouping Boundarys based on Clustering and Statistical Learning, In: Proceedings of the 2014 International Computer Music Conference (ICMC2014), pp. 125–128 (2014)
22. Hamanaka, M., Hirata, K., and Tojo, S.: Music Structural Analysis Database based on GTTM, In: Proceedings of the 2014 International Society for Music Information Retrieval Conference (ISMIR2014), pp. 325–330 (2014)
23. MakeMusic Inc.: Finale, `http://www.finalemusic.com/`