Searching in the Structured Space of the Braille Music

Wladyslaw Homenda, Mariusz Rybnik

Searching in the Structured Space of the Braille Music

Wladyslaw Homenda[1] and Mariusz Rybnik[2]

[1] Faculty of Mathematics and Information Science, Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warsaw, Poland
[2] Faculty of Mathematics and Computer Science, University of Bialystok
ul. Sosnowa 64, 15-887 Bialystok, Poland
homenda@mini.pw.edu.pl, mariuszrybnik@gmail.com

Abstract. This paper presents a method for searching the Braille music information based on the semantic description. The searching method is based on comparison of node's metadata that aggregates subtree's structure of information. Methods for the metadata generation is specific to searching orientation and grammar. We focus on methods for notes' durations allocation as well as key and time distribution that are required in searching and rhythm analysis. We propose a syntax analysis method based on semantic analysis using attribute grammar.
Keywords: syntactic structuring, semantic mapping, data understanding, music representation

1 Introduction

Exploring knowledge encapsulated in the information space needs structuring and agile searching methods. In this paper we remind structuring issues connected with the music information space. Then we propose methods for semantic analysis during parsing process which use attribute grammars. The further part is devoted to a method for semantic searching which uses information incorporated in derivation tree.

Any intelligent processing of natural communication languages requires uncovering structures of raw data. There are different ways leading to structuring. Our interest is focused on employing syntactic and semantic structuring of music information with an emphasis on music notations. It is obvious that raw data without any structuring is of no use in intelligent communication. Otherwise the processing covers some characters from alphabet without any meaning. The aim is to create a generic method to integrate syntactic and semantic structuring of music information. This structuring allows for optimized processing of music information described in different languages including Braille music notation and printed music notation. In this study we bind Braille music notation to printed music notation.

1.1 Syntax

Syntactic structuring of music information is the first stage of the analysis process. We utilize context-free grammars for syntactic structuring of Braille music notation and printed music notation. We continue the discussion of syntactic structuring outlined in [2–5].

Let us recall that we use formal grammars, which are systems $G = (V, T, P, S)$ where: V is a finite set of nonterminal symbols (nonterminals), T is a finite set of terminal symbols (terminals), P is a finite set of productions and S is the initial symbol of grammar, $S \in V$. In general productions can be seen as a finite binary relation $P \subset (V \cup T)^+ \times (V \cup T)^*$. A grammar G is context-free one (CFG) $\iff (\forall p)(p \in P \Rightarrow p \in V \times (V \cup T)^*)$.

Attribute grammar is a context free grammar allowing to tie semantics to syntax during syntax analysis. Attribute grammar attaches so called attributes to each of the nonterminals of the grammar. Such elements are divided into two disjoint sets: synthesized and inherited attributes.

Below a subset of productions for Braille music is presented ([6]). Because of the clarity attributes definitions are omitted. The whole set is specified in [4]

$$
\begin{aligned}
&\langle S\rangle \rightarrow \langle music\_desc\rangle\ new\_line\ \langle music\rangle\\
&\langle music\rangle \rightarrow \langle part\_desc\rangle\ new\_line\ \langle part\rangle\ new\_line\ \langle music\rangle \mid\\
&\qquad \rightarrow \langle part\_desc\rangle\ new\_line\ \langle part\rangle\\
&\langle part\_desc\rangle \rightarrow \langle instruments\_info\rangle\ new\_line\ \langle key\_sig\rangle\langle time\_sig\rangle\\
&\langle part\rangle \rightarrow \langle staff\rangle\ new\_line\ \langle part\rangle \mid \langle staff\rangle\ new\_line\\
&\langle staff\_with\_h\rangle \rightarrow \langle measure\rangle\langle barline\rangle\langle staff\_with\_h\rangle \mid \langle measure\rangle\langle barline\rangle\\
&\langle measure\rangle \rightarrow \langle key\_sig\rangle\langle key\_sig\rangle\langle time\_sig\rangle\ space\ \langle measure\_fm\rangle \mid\\
&\qquad \rightarrow \langle time\_sig\rangle\ space\ \langle measure\_fm\rangle \mid\\
&\qquad \rightarrow \langle key\_sig\rangle\langle key\_sig\rangle\ space\ \langle measure\_fm\rangle \mid \langle measure\_fm\rangle\\
&\langle measure\_fm\rangle \rightarrow \langle voice\rangle\ voice\_div\ \langle measure\_fm\rangle \mid \langle voice\rangle\\
&\langle voice\rangle \rightarrow \langle time\_element\rangle\langle voice\rangle \mid \langle time\_element\rangle\\
&\langle time\_element\rangle \rightarrow \langle note\rangle\langle time\_element\rangle \mid \langle note\rangle\\
&\langle note\_gf\rangle \rightarrow \langle octave\rangle\langle note\_gfo\rangle \mid \langle note\_gfo\rangle\\
&\langle note\_gfo\rangle \rightarrow \langle pitch\rangle \mid \langle interval\rangle \mid \langle pitch\rangle\langle prolongation\rangle\\
&\langle pitch\rangle \rightarrow C1 \mid C2 \mid C4 \mid C8 \mid D1 \mid D2 \mid D4 \mid D8 \mid E1 \mid E2 \mid E4 \mid E8 \mid\\
&\qquad \rightarrow F1 \mid F2 \mid F4 \mid F8 \mid G1 \mid G2 \mid G4 \mid G8 \mid A1 \mid A2 \mid A4 \mid A8 \mid\\
&\qquad \rightarrow H1 \mid H2 \mid H4 \mid H8 \mid rest1 \mid rest2 \mid rest4 \mid rest8
\end{aligned}
$$

Since there is no evidence that Braille music notation is a context-free language, we do not attempt to construct a context-free grammar generating the language of Braille music notation. Instead we use context-free grammars covering the language of Braille music notation. Such grammars will generate all constructions of Braille music notation and some others, which are not valid Braille music constructions. This approach cannot be used to generate Braille music scores or parts neither to check their correctness. However, since we employ context-free grammars for processing scores, which are assumed to be correct, this approach is proven. A discussion on construction of context-free grammars covering printed and Braille music notations is outlined in [5].

The notion of *lexicon* is a fundamental concept of syntax analysis. The concept of lexicon for music notation was introduced in [2] and then was developed in

[4, 5]. *Lexicon* is the space of language constructions, each of them supplemented with possible *derivation trees*, also known as *parsing trees*. *Lexicon* includes relations between items of this space. Such a tree satisfies the following rules: a) it is a subtree of the derivation tree of the whole score, b) it is the minimal tree generating the given language construction, c) the minimal tree can be extended by a part of the path from the root of this tree toward the root of the score derivation tree. Due to the last condition, usually there are many trees for a given language construction.

1.2 Semantics

Languages allow to describe a real world of things, sensations, thoughts, ideas etc. Braille music notation describes the space of hearing sensations, which can be outlined as the space $B \times D \times P$ of triples (b, d, p). Each triple defines the performed sound, where b is the beginning time, d is the duration and p is the pitch of the sound. In general, objects of the real world may be outlined with much richer set of features, but this simple triple is sufficient for our discussion, c.f. [5].

Above-mentioned approach is very generic, it refers to physical essence of a sound and has not any links to a particular type of music description. This structure can be used for any music notation, especially for Braille music notation. The definition of the hearing sensation space is also very useful in case of other structural operations, e.g. *conversion* between different types of music description, c.f. [5].

The purpose of using the world of real objects is to tie meaning to syntactic structures of music descriptions. The idea of collaborating syntactic and semantic has found the practical application to processing of music information. It has been involved in a real processing of Braille music accomplished in the Braille Score project, c.f. [7].

As mentioned in the previous section, descriptions of music notation expressed in different languages and representation of music notation in different formats are cast on the world of hearing sensations. Such casts are called semantics of descriptions and representations of music information. Formally, let B is the lexicon of Braille music notation and H is the space of hearing sensation. Semantics S is a relation:

$$S \subset B \times H$$

The space of language constructions is immersed in the space of sounds. The immersion gives values of real world to language constructions. The immersion defines meaning of language constructions, defines semantics.

2 Music data parsing

Parsing, also known as syntactic analysis, is aimed at structuring raw music data. The result of this process is parsing (derivation) tree compatible with specified grammar. Moreover, the attribute grammar produces semantic data attached to the tree's nodes.

2.1 Key and Time Signature Passage

All of the nonterminals in Braille music grammar are equipped with attributes connected with *key* and time signature. The knowledge of the *key* and time (as synthesized attributes) is discovered in productions and passed to the top of the tree, to the initial nonterminal:

$<fifths> \rightarrow \sharp \mid \flat \mid \natural \mid \sharp\sharp \mid \flat\flat \mid \natural\natural \mid \sharp\sharp\sharp \mid \flat\flat\flat \mid \natural\natural\natural \mid etc.$
$<fifths> .fifth = id$

$<time\_sig> \rightarrow num\_ind\ <beat><beat\_type>$
$<time\_sig> .beat = (<beat> .beatH, <beat\_type> .beatL)$

$<beat> \rightarrow higher\_1 \mid higher\_2 \mid etc.$
$<beat> .beatH = id$

$<beat\_type> \rightarrow lower\_1 \mid lower\_2 \mid etc.$
$<beat\_type> .beatL = id$

Key and *time signature* attributes are passed from the root of the tree to the other structures of the notation as inherited attributes. Each of the productions has a rule similar to this:

$<part_1> \rightarrow <staff>\ new\_line\ <part_2> \mid <staff>\ new\_line$
$<staff> .beat = <part_1> .beat$
$<part_2> .beat = <part_1> .beat$

This method allows to distribute *key* and *time signature* info in structured space of music. *Key* and *time* knowledge flows from *key/time signature* branch to the root of the tree and then it is propagated to the lower parts of the tree.

2.2 Notes' Duration Identification and Passage

In case of Braille music notation, because of limited number of available characters, notes of the duration x and $\frac{x}{16}$ share the same symbol, e.g. *whole note C* and *sixteenth note C* are represented by the same Braille cell. This ambiguity demands contextual analysis to investigate the correct duration of the note.

This issue is a nondeterministic problem and it involves voices, because voices are fully filled in time dimension.

Simulation tree method One of the practical solutions of this nondeterministic issue is simulation tree. We assume, that the sound of each note lasts from the beginning of the note to its end, i.e. no articulation, ornamentation or dynamic has influence on the duration of the sound. This tree is defined as follows (lets be $N = (n_0, n_1, ..., n_k)$ - notes in voice and $S = (s_0, s_1, ..., s_{k+1})$ - time slots between neighboring notes, i.e. s_0 is the time when first note starts, s_1 – the time when first note ends and the second starts, etc.:

- nodes of the tree are labeled with elements from S
- root of the tree is labeled with s_0

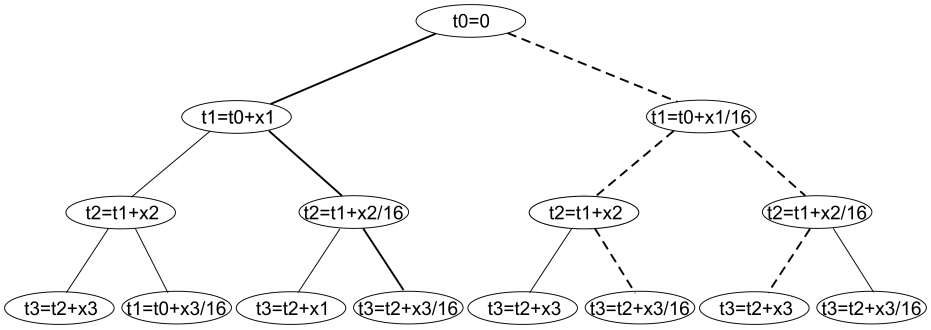

Fig. 1. Simulation tree draft for calculating duration

– each node labeled with s_x has two children labeled with s_{x+1}, that represent two variants of the note duration n_x

The draft of simulation tree is given in the Fig. 1.

Such definition describes all possible time allocations. Each of the leaf of the simulation tree represents unique time allocation in the voice, because leaf in the tree is labeled by s_{k+1} - the space after last note in the voice. Incorrect time fillings are easily detectable. Braille music definition demands unambiguous situations in case of notes duration, i.e. there is only one path from the root to the leaf in the simulation tree that is filled in correct way. Fig. 1 presents simulation tree for 3 notes. Let assume that notes may represent the same durations, e.g. C, D, and E whole/sixteenth notes (whole/sixteenth are described by the same Braille cell), and time signature has a proper value. This is a case that fits the scheme ,,one left and two rights", which leads to three concretizations: left-right-right, right-left-right and right-right-left (marked in the Fig. 1 as bold solid and dotted lines). Each of above mentioned filling schemes fills voice in correct manner but only one of them is desired and it is left-right-right (marked as bold, solid line), because in other case there should appear special character after s_1(in the right part of the tree, where starts two dotted, bold lines; Braille specific). That character is to mark explicitly the duration of the note. Appearance of such Braille signs excludes some durations of the note and causes the tree to loose some paths. All paths are present in the tree in the Fig. 1, so there is no duration mark in the notation.

Attribute flow method We propose method that bases on attribute grammar. Attribute grammars for music notation was introduced in [3]. In this paper we use concepts form this paper to resolve the problem of time/duration ambiguity of Braille music notation. In fact this method creates implicit simulation tree during syntax analysis.

Attribute flow method requires two stages (two attributes flows) during processing. Also we assume that all nonterminals located below ,,voice" element in

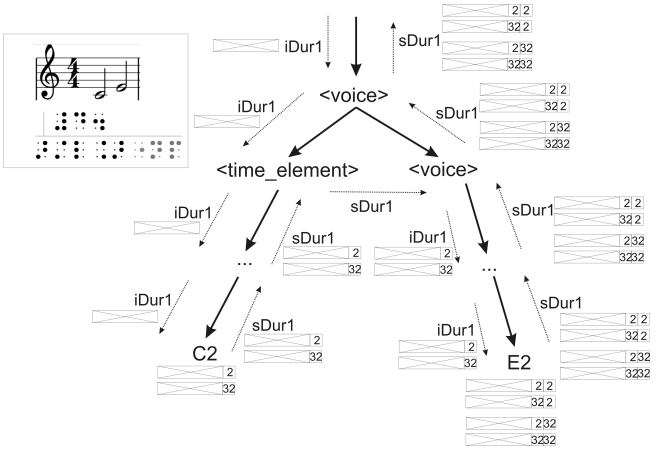

Fig. 2. Attribute flow during the first stage for production
<voice>→<time_element> <voice>

derivation tree are supplemented with four attributes: two synthesized (*sDur1* and *sDur2*) and two inherited (*iDur1* and *iDur2*).

First stage During first stage the attribute *iDur1* is passed down to the leaf, change in *sDur1* is attempt according to semantic rule, *sDur1* is passed toward the root to the ,,time_element" nonterminal, it changes *iDur1*, then *iDur1* is passed down to the leaf, ... This procedure lasts till the end of the notes in current voice. The example of attribute flow is presented in the Fig. 2

It allows to collect all possible variants of voice's time allocation. After that phase, when *sDur1* is synthesized in ,,voice" nonterminal, the correct variant is chosen. According to remark in 2.2 paragraph, thanks to Braille music definition, there exists one unique time allocation. This one particular solution is assign to *iDur2* attribute of ,,voice" nonterminal.

$<note\_gfo>\rightarrow<high_1> \mid <interval> \mid <high_2><prolongation>$
$<high_1>.iDur1 = <note\_gfo>.iDur1$
$<note\_gfo>.sDur1 = <high_1>.sDur1$
$<high_2>.iDur1 = <note\_gfo>.iDur1$
$<prolongation>.iDur1 = <high_2>.sDur1$
$<note\_gfo>.sDur1 = <prolongation>.sDur1$

$<high>\rightarrow Nx$, where $Nx = (N, x) \in \{C, D, E, F, G, A, H, rest\} \times \{1, 2, 4, 8\}$
$<high>.sDur1 = (<high>.iDur1 \cup \{x\}, <high>.iDur1 \cup \{16x\})$

$<prolongation_1>\rightarrow * <prolongation_2>$
$<prolongation_2>.iDur1 = <prolongation_1>.iDur1 \cup \{\frac{1}{2}\}$
$<prolongation_1>.sDur1 = <prolongation_2>.sDur1$

$<prolongation> \rightarrow *$
$<prolongation>\ .sDur1 = <prolongation>\ .iDur1 \cup \{\frac{1}{2}\}$

$<measure\_fm> \rightarrow <voice>\ voice\_div\ <measure\_fm> \mid <voice>$
$<voice>\ .iDur1 = ()$
$<voice>\ .iDur2 = \text{chooseCorrectTimeAllocation}(<voice>\ .sDur1, <voice>\ .beat)$

$<time\_element_1> \rightarrow <note_1> <time\_element_2> \mid <note_2>$
$<note_2>\ .iDur1 = <time\_element_1>\ .iDur1$
$<time\_element_1>\ .sDur1 = <note_2>\ .sDur1$
$<time\_element_2>\ .iDur1 = <time\_element_1>\ .iDur1$

$<voice_1> \rightarrow <time\_element_1> <voice_2> \mid <time\_element_2>$
$<time\_element_1>\ .iDur1 = <voice_1>\ .iDur1$
$<voice_2>\ .iDur1 = <time\_element_1>\ .sDur1$
$<voice_1>\ .sDur1 = <voice_2>\ .sDur1$
$<time\_element_2>\ .iDur1 = <voice_1>\ .iDur1$
$<voice_1>\ .sDur1 = <time\_element_2>\ .sDur1$

Second stage After the last note, the correct time allocation is chosen and is propagated in the tree. It uses *iDur2* and sDur2 attributes in the same way the *iDur1* and *sDur1* were used in the first stage. In this phase each note cuts one proper time duration and assigns to itself. After the second stage, the notes have allocated correct durations.

General remarks Both methods are practical solutions of the nondeterministic problem of the note's time allocation. Simulation tree and attribute flow method are equivalent. In fact, the attribute flow method is implicit simulation tree method. It creates simulation tree during the process of generation of the all possible time allocations. Simulation tree is ,,orthogonal" to derivation tree, as it is shown in the Fig. 3.

There is a possibility to optimize proposed method of attribute flow by saving memory space. Optimization requires change in semantic rule for $<high>$ production. The method will collect only one sign for each note during the first attribute flow (instead of duplicating the previous formulas):

$<high> \rightarrow Nx$, where $Nx = (N, x) \in \{C, D, E, F, G, A, H, rest\} \times \{1, 2, 4, 8\}$
$<high>\ .sDur1 = <high>\ .iDur1 \cup \{x\}$

After applying the changes above the simulation tree is created in function ,,choose" and still allows to investigate the one, possible solution. It allows however to save memory during attribute flow, as original variant has space complexity $O(|N| \cdot 2^{|N|})$ and optimized method has space complexity $O(|N|)$, where $|N|$ is the number of notes in voice.

The mentioned methods are applicable even if there is a need to check $2^{|N|}$ possible time allocations because of the fact that one voice does not contain huge numbers of notes. The average number of the notes in the one voice is about 10.

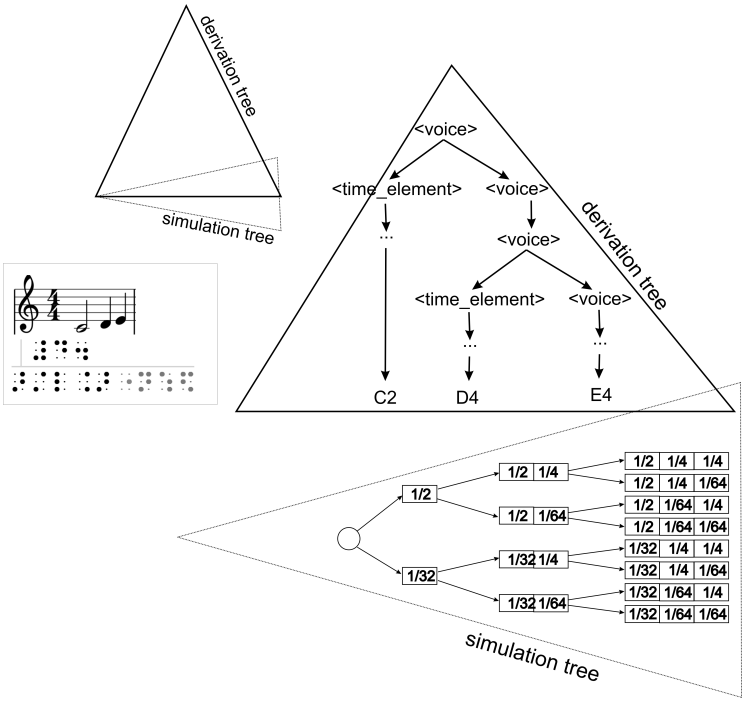

Fig. 3. Simulation and derivation tree in the attribute flow method

3 Selection and Semantic Searching operation performed on the Music Notation

This paragraph contains a description of selection and searching operations. We discuss here a different types of grammars and their influence on selection performed in the structured space of music. Selection is a prelude to searching operation.

In this case we focused on rhythmic analysis preceded by notes' durations allocation. In previous paragraph we showed method for duration allocation as well as *key* and time signature distribution, which are crucial for rhythmic analysis in Braille music notation. Other types of searching information are easy to perform in similar way – as attributes flow during syntax analysis or in the space of sounds.

3.1 Different types of Grammars and Selection Operation

Music information is a data that can be presented in many different formats, e.g. printed music, Braille music, MIDI, etc. Each type is described by specified grammar, in this case they are approximated by context-free grammars. Moreover, in frames of one format there exist many grammars. Each of these

grammars defines the same language (data format), but exposes different property of the language. The examples of such grammars are given in [3], where authors introduced two grammars for printed music: ,,time-prior-to-pitch" and ,,pitch-prior-to-time". [3] illustrates selection performed in the structured space of printed music notation.

Selection made in the notation space leads to selecting some nodes in the derivation tree. The structure of selected nodes in the tree differs depending on the grammar type. Some grammars make nodes to be consistent, i.e. there is some node that is an ancestor for all selected nodes and that node does not have unselected children as descendant. In other words, there exists one node, that derived all and only selected notation.

Some language has natural orientation, e.g. Braille music is voice oriented, i.e. selection of the notes in one voice is going to generate consistent selection in derivation tree. Other orientations for Braille music are also possible. In case of the printed music, in the paper [3] were given grammars: pitch and time oriented.

3.2 Semantic Searching

Semantic search allows to search for all occurrences of notation described in semantic way, e.g. all fifth accords. Syntax searching, where algorithm looks for the exact occurrence of the searching text, is a special case of semantic search.

During searching operation the lexicon is used.

Let us assume that grammar type and selection operation generate consistent selection of nodes in the derivation tree. For each lexicon's element notation is described as data put in the root of the subtree. That description should be connected with the type of grammar and searching operation. Method for accord description, in case of interval analysis, were given by Allan Forte in [1]. Descriptions are applicable during syntax analysis thanks to attribute flow.

To search specified notation described by some semantic, all that should be done is searching nodes in the whole derivation tree. Lexicon contains many elements for one piece of notation. The difference between these elements is in subtrees – each subtree has different height. The bigger height, the more precise description it represents. Searching for a *note* which description is aggregated in nonterminal ,,<*note*>" means that any of such *notes* in the whole measure meets the condition. If searching for a *note* involves nonterminal ,,<*measure*>" it means that all occurrences of specified *note* in that particular time slot in all *measures* of the score are desired. Finally, nonterminal ,,<S>" defines exactly one occurrence of the searching *note*.

Semantic search requires semantic which is delivered during syntax analysis as floating attributes or as the world of hearing sensation (thanks to semantic mappings).

3.3 Semantic Searching Example

Fig. 4 presents an example of semantic searching in the structured space of Braille music. The task is to find all occurrences of the accords that have the

same intervals and duration. Please note that Braille music allows to denote accords in more than one manner (c.f. [6]), so our task is not an usual (syntax) search. Grammar presented in [4] allows for such operation. We tie with all ,,$< time\_element >$" nonterminals a vector that specifies desired information about derived structures. That activity can be done after allocation of notes' duration and *key/time signatures* passage in the derivation tree. The vector is labeled as $[d; x_1 - x_2 - ... - x_n]$ where d is the note (accord) duration, $n+1$ is the number of noteheads in the accord, x_k is the interval between $(k-1)$-th and k-th noteheads.

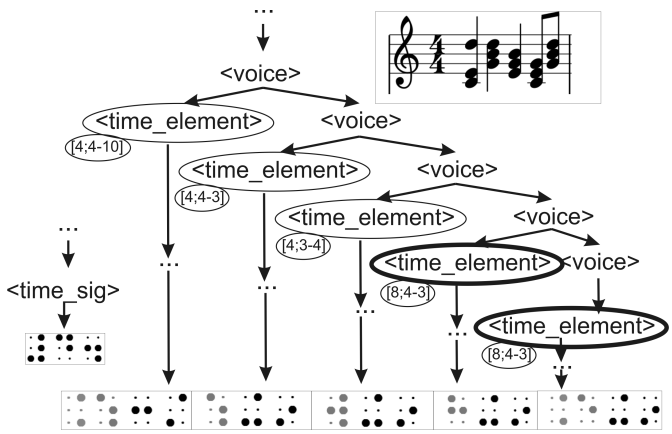

Fig. 4. Semantic Searching Example

Let us assume that we want to find all *eighths* accords built from intervals 4 and 3, so the vector is $[8; 4-3]$. To fulfill this task all that should be done is vector (,,<*time_element*>" nonterminal metadata) comparison. No syntax of the score is touched. The method points two occurrences, marked in the Fig. 4 with bold ellipses. The result of the operation is a piece of notation that is derived from the pointed nonterminals marked with bold ellipses.

4 Conclusions

This paper presents approach for searching in the structured information space of Braille music. Searching is understood as localization of specified node in the derivation tree, which corresponds to node of the subtree paired with the searched notation in the lexicon space. Our approach is rhythmic oriented, i.e. we investigate mutual time dependencies between notes. We needed to allocate duration for the notes to cope with that.

The direct expansion can be aimed at providing another methods for aggregating data at the subtree's root. The important issue is to connect with intelli-

gent human-computer interface developing because conditions imposement is a bottleneck for the searching in the structured space of music data.

Acknowledgment

The research is supported by the National Science Center,
grant No 2011/01/B/ST6/06478, decision no DEC-2011/01/B/ST6/06478.

References

1. Forte, Allen (1973). The Structure of Atonal Music. New Haven and London: Yale University Press. ISBN 0-300-01610-7 (cloth) ISBN 0-300-02120-8 (pbk).
2. Homenda W., Rybnik M.: Querying in Spaces of Music Information, Lecture Notes in Artificial Intelligence, LNAI 7027, pp. 243255, Springer-Verlag Berlin Heidelberg, 2011
3. Homenda W., Rybnik M.: Knowledge-Driven Syntactic Structuring: the Case of Multidimensional Space of Music Information, S.W. Liddle et al. (Eds.): DEXA 2012, Part I, Lecture Notes in Computer Science, LNCS 7446, pp. 438452, Springer-Verlag Berlin Heidelberg, 2012
4. Homenda, W., Sitarek, T.: Performing Operations on Structured Information Space of Braille Music, Andreas Konig et al. (Eds.) KES 2011. LNCS (LNAI), vol. 6884, pp. 232–241. Springer, Heidelberg (2011)
5. Homenda, W., Sitarek, T.: Notes on automatic music conversions, M. Kryszkiewicz et al. (Eds.): ISMIS 2011, LNAI 6804, pp. 533–542, 2011.
6. Krolick, B.: How to Read Braille Music, 2nd edn. Opus Technologies, 1998
7. Grant no N R02 0019 06/2009, Breaking accessibility barriers in information society. Braille Score - a computer music processing for blind people, Institute for System Research, Polish Academy of Sciences, report, Warsaw, 2011.