Efficient Processing the Braille Music Notation

Tomasz Sitarek, Wladyslaw Homenda

Efficient Processing the Braille Music Notation

Tomasz Sitarek and Wladyslaw Homenda

Faculty of Mathematics and Information Science
Warsaw University of Technology
Plac Politechniki 1, 00-660 Warsaw, Poland
Tomasz.Sitarek@ibspan.waw.pl
http://www.mini.pw.edu.pl/~homenda

Abstract. Problem stated here is connected with music information processing, especially with Braille music notation. The main objective of this paper is usage of semantics for optimization of Braille music scores processing. This issue is important in the area of huge Braille music scores. Our approach is based on structuring – both syntactical and semantic – in spaces of music information. Optimization would not be possible without such structuring. The main idea is connected with logical score partitioning into smaller pieces that are weakly dependent between each other. Optimization is based on closing changes to small syntactical and semantic items of the structure. Each change during editing touches on of such small items instead of processing significant parts of the whole structure.
Keywords: data understanding, knowledge processing, music representation, music data processing

1 Introduction

In this paper we discuss the problem of processing Braille music notation. Braille music notation is an example of language of communication between people and, in this case, it is called a language of *natural communication*, c.f. [2, 6]. Communication is seen as intelligent exchange of information, which involves information understanding by all sides of communication. Nowadays technologies support information processing helping to improve interpersonal communication. Languages of communication can be seen as tools for constructing vehicles carrying information. These vehicles are texts of natural languages, scores of music notation, scores of Braille music. This remark brings a way to deal with information processing including personal and automatic processing. Such processing requires information structuring as well as identification of structures of information.

In this study we discuss syntactic and semantic structuring of Braille music notation. The discussion is aimed on optimization of automatic processing of big scores of Braille music notation. Namely, we investigate a possibility of avoiding processing of big parts of the whole score in editors of Braille music. Such optimization is worth attention when big scores are edited. However, due to properties of music information, for many symbols of music notation, simple editor's tools do not allow for limited processing. It is necessary to employ

deep syntactic and semantic analysis of the score or its fragment corresponding to editing activities in order to limit to necessary minimum the spectrum of processed symbols.

The paper is structured as follows. Fundamentals of syntactic and semantic tools involved in structuring music information are recalled in section 2. In section 3 we describe shortly formats of music information. We only recall general purpose music representation formats and describe to some extent formats related to Braille music notation. These formats are crucial for the main track of this study. Introduced efficiencies in Braille music processing are studied in section 4. On the basis of illustrative examples we describe syntactic and semantic methods leading to minimization of parts of Braille music score subjected to processing during editing operations. Expansion of presented examples to other symbols of music notation is straightforward. Finally, we come to conclusions anchored in real Braille music processing editor developed in frames of the acknowledged research project.

2 Syntactic and Semantic Mappings

Any intelligent processing of constructions of languages of natural communications 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 special emphasis put on Braille music notation. It is obvious that raw data without any structuring is useless in intelligent communication. Otherwise the processing covers some characters from alphabet without any meaning. The aim is to create generic method for 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. For people with good eyesight we bind Braille music notation with printed music notation in this study. The method is an extension of a likewise study in [4].

2.1 Syntax

Syntactic structuring of music information is the first stage of the analysis process. We will utilize context-free grammars for syntactic structuring of Braille music notation and printed music notation. We refer to and will continue discussion of syntactic structuring outlined in [4].

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) $\iff (\forall p)(p \in P \Rightarrow p \in V \times (V \cup T)^*)$.

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 in generating Braille music scores or parts of scores or in checking their correctness. However, since we employ context-free grammars for processing scores, which are assumed to be correct, the approach is proven. A discussion on construction of context-free grammars covering printed and Braille music notations is outlined in [4].

2.2 Lexicon

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:

- it is a subtree of the derivation tree of the whole score,
- it is the minimal tree generating the given language construction,
- 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. We do not recall the meaning of parsing tree in a context-free grammar, refer to [5] for definition of it.

Different trees supplementing a given language construction describe different context of the language construction. For instance, if we consider a sequence of consecutive notes, the minimal derivation tree for these notes matches all such sequences in the whole score, if more than one is present. If the minimal tree is extended to the root of the derivation tree of the whole score, than it represents only this given sequence of notes, c.f. [3] for details. The concept of the lexicon can be applied, for instance, for better understanding and better performing of structural operations, e.g. *find* operation.

2.3 The World: the Space of Hearing Sensation

Languages allows 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 beginning time, d is duration and p is pitch of this sound. In general, objects of the real world may be outlined with much reacher set of features, but this simple triples are sufficient for our discussion, c.f. [4].

Above mentioned approach is very generic, refers to physical essence of a sound and has not any links to a particular notation. This structure can be used for any music notation, especially Braille music notation. This definition of the space of hearing sensation is also very useful in case of other structural operations, e.g. *conversion*, c.f. [4].

The purpose of using the world of real objects is to tie meaning to syntactic structures, i.e. to tie meaning to lexicon elements of the Braille music notation. This assumption allows us to cast different descriptions music information and different formats representing music information onto the space of hearing sensation. In this way, it is possible to construct collaborating methods, which operate on these different descriptions and formats, c.f. [1]. We apply the idea

for formats used in a real processing of Braille music accomplished in frames of the Braille Score project, with the BMF format described in next sections as the space of music sensations.

2.4 Semantics

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 Braille music notation is tightly connected with objects in internal format. Here we consider the BMF format as the example. Every element of the Braille music notation (a character, a Braille cell) contains information about its owner – element from BMF. It is obvious that every Braille cell has such element, because:

- characters present in notation **represent** some data from BMF. Otherwise they are redundant and would be thrown away from notation,
- characters in notation **create** some data in BMF. Otherwise they are redundant, caries no information and would be thrown away from notation

Every element in notation has related element(s) in BMF, and every element in BMF has related element(s) in notation. This relation is many-to-many. One element from BMF may create many Braille cells and one Braille cell may create many elements from BMF. Nevertheless, one meaning is chosen, often not at once, but depending on neighboring Braille cells.

3 Formats Description

Musical data is stored as files or physical scores (manuscripts or printed ones – images/photos of scores). The files are written in different formats. Rough division enumerates formats that are printed score (e.g. MusicXML files), Braille score (files with Braille music) or pure sound (e.g. MIDI files) oriented.

When processing music data it is convenient to use more than one internal format. Internal formats should meet external formats of data or to make processing easier.

A set of the internal formats is presented in the Figure 1. These structures store musical information in varied way: external types, printed music oriented, Braille music oriented and Braille music files. Between the most of these types is available bidirectional conversion, except one way conversion BNMF $\rightarrow$ BN. There exist implicit conversions BN $\rightarrow$ ASCII $\rightarrow$ BNMF.

Bubble ,,ASCII" represents Braille music written as ASCII file. Bubbles ,,LONG" and ,,PAGE" indicate displayed notation in two manners: as infinite line or as broken line. These two bubbles do not handle structured data, in opposite to other bubbles. Each of the other bubbles from the Figure 1 is described in next paragraph.

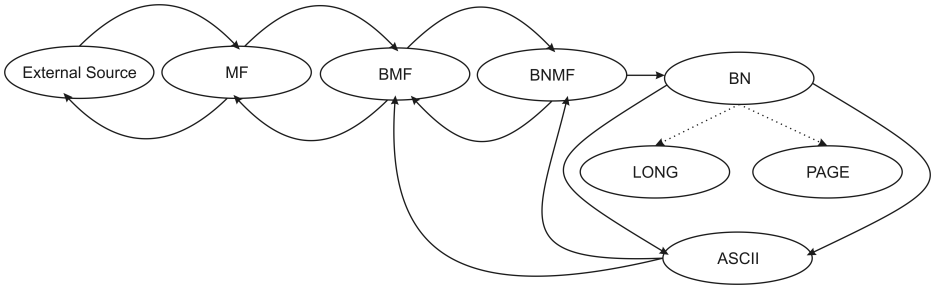

Fig. 1. A typical data flow in Braille Score

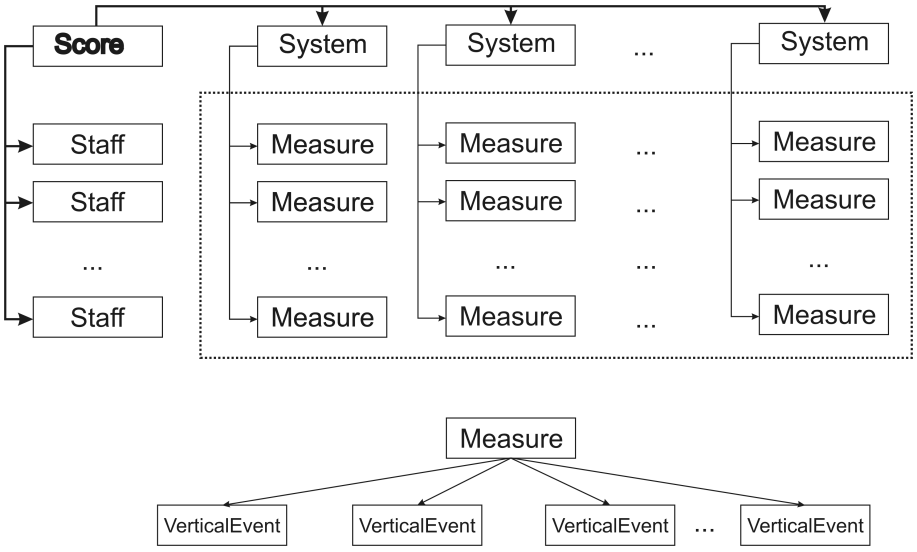

Fig. 2. MF

3.1 External Source

This data source is used to create the main internal format called ,,MF". There is possibility to build MF from following sources:

- MusicXML – a popular file format used to write music scores (c.f. [7]),
- MIDI – file format that contains only sound information (c.f. [8]),
- .bmp, .jpg, .tif, .png – images of scores, which after recognition process will create MF,

3.2 MF

MF (Figure 2) is the main internal format. It is not tied with Braille notation. MF can be displayed and edited as printed music notation. MF consists of several abstract data types, the main are:

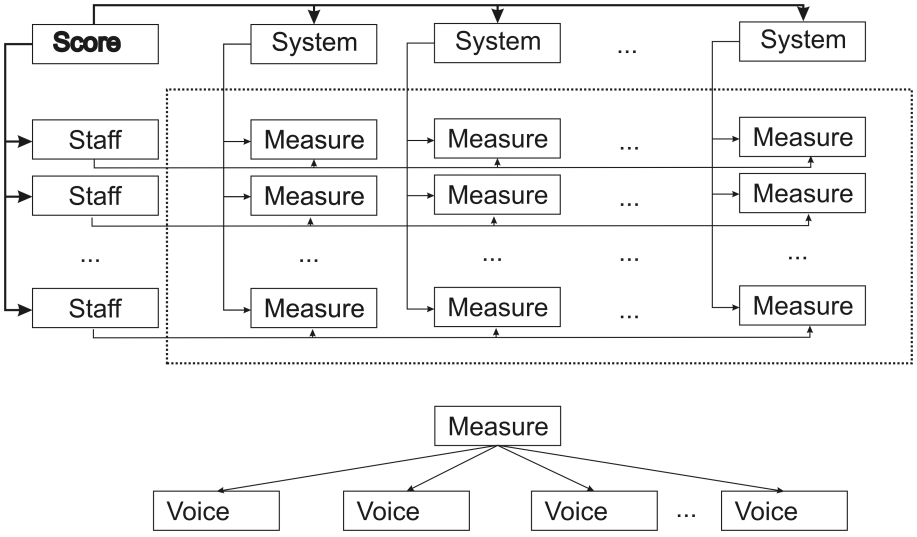

Fig. 3. BMF

- **Score** – represents whole score, contains **Staffs**, **MusicSystems**, time signature and many other information about page parameters and metadata (such as title, author, composer)
- **Staff** – contains information about instrument, key signature and clef
- **Music System** – contains barlines information and **Measures** (this construction can be misleading because of different naming convention: sometimes multiple staves is called measure; in Braille Score each measure contains one staff, many staves creates system)
- **Measure** – contains **VerticalEvents**
- **VerticalEvent** – elements that represents one particular event during given measure's time, contains voices slots and additional dynamic/ornamentation information; each voice slot may contain note/rest element

This above description is general draft of MF structure. Mentioned elements are emblazoned by additional properties, such as ornamentation, articulation, dynamic symbols.

3.3 BMF

BMF (Figure 3) is the main internal format in Braille music editor. It handles Braille music information encapsulated at high abstraction level. This format's structure is similar to ,,MF", but there are some differences in regard of Braille music notation. BMF consists of:

- **Score** – represents whole score, contains **Staffs**, **Systems**, time signature and many other information about page parameters and metadata (such as title, author, composer, interval reading direction)
- **Staff** – contains **Measures**, instrument names

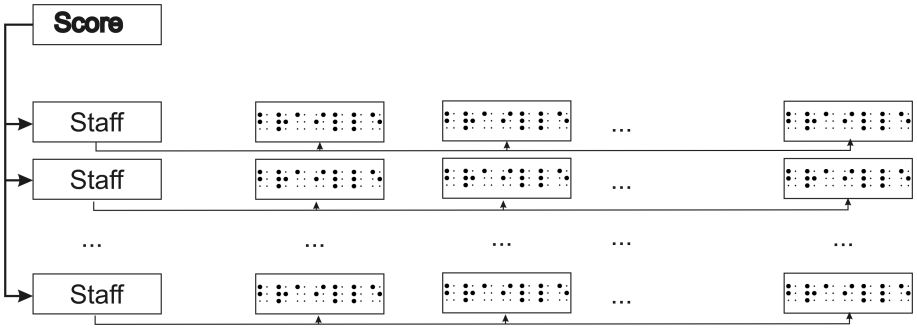

Fig. 4. BNMF

- **System** – contains **Measures**, barlines, time signatures
- **Measure** – contains **Voices**, key signature
- **Voice** – contains notes/rests

3.4 BNMF

BNMF (Figure 4) is simple format that represents Braille dots at high level abstraction with semantic attached. This format is organized as follows:

- **Score** – represents whole Braille music score, contains **Staffs**
- **Staff** – contains **Measures**
- **Measure** – contains **BSymbols** and ending **Small Context**
- **BSymbol** – element that represents Braille dots
- **Small Context** – set of elements, that allow to process notation more efficiently, will be described in the next paragraph

Each BSymbol has information about protuberant dots of this cell. It has connection to element from BMF, which produces this Braille cell and gives the meaning to it.

This format was chosen to be the target of editing operations. Changes in Braille music notation are made in this format. Then, if necessary, changes are propagated in other formats.

3.5 BN

BN (Figure 5) is an intermediate layer, that contains both: Braille cells with semantics as an infinite line and Braille cells with semantics as a broken line. This format is an element, that can be displayed or written into the file in both manners: as an infinite line or as a broken line.

3.6 Context

Context is used when reading Braille music notation (e.g. from ASCII file or from BNMF format). It contains information about signs read previously which has influence on current sign.

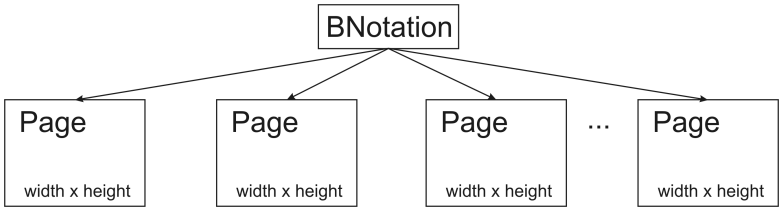

Fig. 5. BN

Elements of Context have different scope. Arbitrary division itemizes inter-measure elements (33 units) and extern-measure elements (7 units). The first ones have scope limited to one measure, i.e. they have no influence on signs from other measures. Extern-measure elements have scope limited to one staff and will be called ,,Small Context". To sum up: Context consists of Small Context and inter-measure elements.

Small Context contains following elements:
- opened slurs count
- opened wedges count
- intervals reading direction
- slurs numbering
- short slurs error check
- last read note (pitch and octave)
- last read attributes (key and time signature)

Small Context Equality Two Small Context are equal if and only if theirs respective fields are equal.

4 Efficient Processing the Braille Music Notation

Very often editing operation changes only one measure, and that operation has small scope – single measure or a few measures in the neighborhood. There is no need to process whole score or staff. Using the Small Context, there is an ability to avoid expensive computing.

4.1 Draft

Thanks to Context construction, i.e. Small Context included in Context, notation can be faster processed. All changes are made on BNMF, because of presence of pure notation in this case, and operation touches directly notation.

Because of Lexicon construction (rejection from consideration some structures, e.g. slurs between staves), any of the staves can be process independently. Context is used during processing every measure of the score. Each measure remembers Small Context that occurred at the end of processing this measure. Small Context – because it is a subset of Context and contains all extern-measure data. Remembering of the ending (except beginning) Small Context because:
- for each measure k, except the first measure, the beginning context is the ending context of measure (k-1)
- for measure number 0 the beginning context is new context
- knowledge of ending state at the end of staff

4.2 Mechanism of Processing

Processing the Braille music notation occurs in two situations:

- processing a notation the first time – no knowledge about ending contexts for the measures
- processing a notation again – knowledge about ending contexts for the measures because of previous processing

The first case does not benefits from this method for efficient processing. Processing the whole notation is all and enough what can be done.

The second case allows to avoid processing whole notation. This situation occurs when notation has been changed; the changed measures are marked. We assume that change affects only one measure. In other case, we can treat multiple position change as multiple changes.

Algorithm 1 Algorithm for efficient score processing

```
 1: for all s in Staves do
 2:    ctx = new Context
 3:    for all m in s.Measures do
 4:       if m.HasChanged OR m.SmallContext == NIL then
 5:          if m != s.Measures[0] then
 6:             ctx = s.Measures[m.Number-1].SmallContext
 7:          end if
 8:          process measure m with Context ctx
 9:          m.HasChanged = false
10:          if m.Number < s.Measures.Count - 1 AND !ctx.Equals(m.SmallContext)
             then
11:             s.Measures[m.Number+1].HasChanged = true
12:             m.SmallContext = ctx
13:          else if m.Number == s.Measures.Count - 1 then
14:             m.SmallContext = ctx
15:          end if
16:       end if
17:    end for
18: end for
```

4.3 Deferred Semantic Bounding

In paragraph 4.2 was made assumption, that each change is processed separately. That approach causes sometimes correct but inefficient processing.

There is a need to design a special mode which allows to defer semantic bounding. This mode prevents score processing till it become switched off. Just after that the score is processed and semantic is bounded too.

The mentioned above mode is useful when paired symbols included in Small Context are added, i.e. slurs or wedges beginnings and endings. It is worthwhile to process score after full symbol insertion, i.e. beginning and ending symbol.

Otherwise it causes instability at the end of measure where insertion occurs: Small Contexts are not equal to ctx. This instability is propagated from the

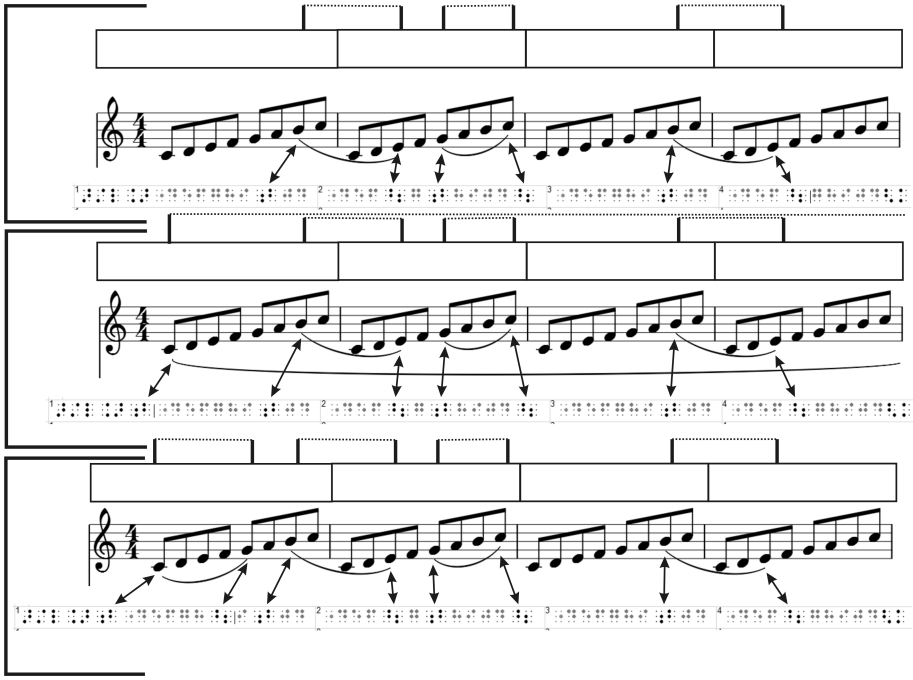

Fig. 6. General description of the editing activity.

changed measure to the end of the staff (in pessimistic situation). The insertion of the second of the paired symbol causes the same: processing remaining measures. The second phase reverts almost all changed ending Small Contexts to the state from before the first processing.

To sum up:

- **with** deferred semantic bounding the processing affects a few required, changed measures
- **without** deferred semantic bounding the processing may affect whole staff twice in pessimistic case (insertion of paired symbols and processing them separately)

Deferred semantic bounding example This section presents example to illustrate deferred semantic bounding issue. Figure 6 shows editing operation that is aimed to add slur in the notation. This process is partitioned in three stages: before operation, after inserting beginning sign of the slur and after inserting ending sign of the slur. Each phase of the slur insertion is described in sequence by: general overview draft, printed music notation and Braille music notation. Correspondence between beginnings and endings of slurs in printed music and signs in Braille music is marked in Figure 6 by bold arrows.

The initial score contains three slurs (first stage). In the second stage beginning of the slur is added to the first note. That makes the inserting slur to be opened from one of the ends. In terms of processing the Braille music it creates slur that never ends, that lasts till the score ends.

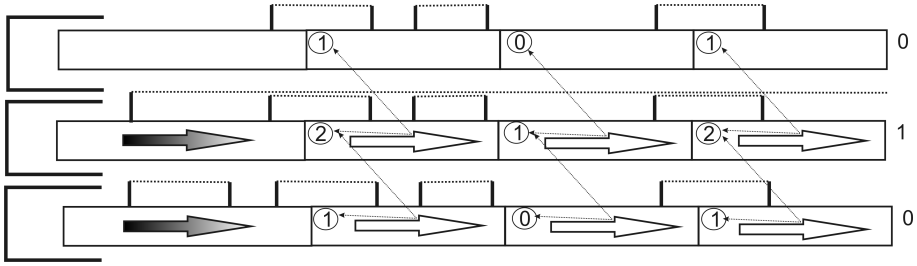

Fig. 7. Inefficient processing during edition.

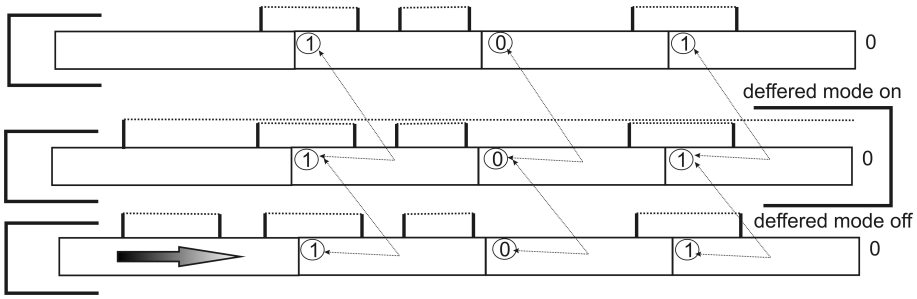

Fig. 8. Efficient processing during edition.

Ending of the slur is inserted in the third stage. After this activity the slur is defined by beginning and ending. The score has four slurs. All changes touch one measure. It means that it is enough to process this particular measure. Efficient processing is performed in that case thanks to deferred semantic bounding.

Figure 7 illustrates inefficient processing. Slurs are marked in the same way as in the Figure 6 – by vertical, bold segments connected in pairs by dotted lines. At the end of each measure the number of opened slurs in marked.

The big arrow in the measure rectangle indicate processing this measure (bounding semantic). The big and dark arrow is connected with changed measure, which should be always processed because it contains the change and is the smallest item of syntactical and semantic sturcturing. The light arrow matches processing that occurs in the measure as a result of changes of small context (refer to algorithm 4.2, step 10). The light arrows appear because of absence of the deferred semantic bounding. In all that cases occur difference between number of opened slurs in first to second and second to third stages for respective measures.

Please notice that the light arrow processings are redundant because numbers of opened slurs in the third stage are the same as in the first stage. That means no final change in opened slurs in measures marked with light arrow.

The above conclusion brings as to deferred semantic bounding. This term means that notation is not processed. This implies no changes in small contexts. Finally, this implies no processing in measures, except one measure where editing operation was performed.

Correct processing is shown in the Figure 8. The dark arrow occurs once because of insertion the beginning sign for slur in deferred semantic bounding mode. There is no light arrows.

5 Conclusions

This paper is a description of the idea of efficient processing Braille music notation. Appropriate algorithm was proposed to show described method and illustrate additional consequences such as deferred semantic bounding. Efficient processing way can be used with other music languages, e.g. printed music notation. It demands only small context definition and smallest syntax and semantic structures selection.

Introduced method and deferred semantic bounding mode are implemented in Braille Score – application that allows to process music data. Nowadays deferred semantic bounding mode is activated manually, but we consider to automatize this feature.

Acknowledgement

This work is supported by The National Center for Research and Development, Grant no. N R02 0019 06/2009.

References

1. Breaking accessibility barriers in information society. Braille Score - design and implementation of a computer program for processing music information for blind people, the reserach projectconducted in the Systems Research Institute, Polish Academy of Sciences, 2009-2012, supported by The National Center for Research and Development, Grant no. N R02 0019 06/2009
2. Homenda W.: Integrated syntactic and semantic data structuring as an abstraction of intelligent man-machine communication, ICAART - International Conference on Agents and Artifcial Intelligence, Porto, Portugal, pp. 324-330, 2009
3. Homenda W., Rybnik M.: Querying in Spaces of Music Information, Lecture Notes in Artificial Intelligence, LNAI 7027, pp. 243-255, Springer-Verlag Berlin Heidelberg, 2011
4. Homenda, W., Sitarek, T.: Notes on automatic music conversions, M. Kryszkiewicz et al. (Eds.): ISMIS 2011, LNAI 6804, pp. 533–542, 2011.
5. Hopcroft J. E., Ullman J. D., Introduction to Automata Theory, Languages and Computation, Addison-Wesley Publishing Company, 1979, 2001
6. Krolick B.: How to Read Braille Music, 2nd edn., Opus Technologies (1998)
7. Castan G., Good M., Roland P.,: Extensible Markup Language (XML) for Music Applications: An Introduction. in: Walter B. Hewlett and Eleanor Selfridge-Field (Eds.) The Virtual Score: Representation, Retrieval, Restoration, Cambridge, MA: MIT Press, pp. 95-102, 2001
8. MIDI 1.0, Detailed Specication, Document version 4.1.1, February 1990.