A Way to Separate Knowledge From Program in Structured Document Analysis: Application to Optical Music Recognition*

Bertrand Coüasnon and Jean Camillerapp
IRISA / INSA-Département Informatique
20, Avenue des buttes de Coësmes
F-35043 Rennes Cedex, France
couasnon@irisa.fr

Abstract

Optical Music Recognition is a form of document analysis for which a priori knowledge is particularly important. Musical notation is governed by a substantial set of rules, but current systems fail to use them adequately. In complex scores, existing systems cannot overcome the well-known segmentation problems of document analysis, due mainly to the high density of music information.

*This paper proposes a new method of recognition which uses a grammar in order to formalize the syntactic rules and represent the context. However, where objects touch, there is a discrepancy between the way the existing knowledge (grammar) will describe an object and the way it is recognized, since touching objects have to be segmented first. Following a description of the grammar, this paper shall go on to propose the use of an operator to modify the way the grammar parses the image so that the system can deal with certain touching objects (*e.g. *where an accidental touches a notehead).*

1 Introduction

If an analysis is carried out on a document with a tight structure, it is important to use a priori knowledge in order to improve the recognition process. However this information is usually not used to a sufficient extent and where it is used it is introduced at different points in the program. This results in a lack of unity in the knowledge, making it difficult to scale up systems and to modify rules. This lack of knowledge use and unity can be explained by the discrepancy between the way knowledge can describe an object and the way this object have to be recognized. This discrepancy is usually linked to segmentation problems.

In Optical Music Recognition, the first segmentation problems encountered are caused by the staff lines connecting all the musical objects. Moreover, complex scores usually have a large number of notes and are polyphonic (different voices on a single staff) such that the information is extremely dense. As a result, some difficulties may be encountered in the segmentation of the document (*e.g.* graphical objects may touch when logically they should not). According to [2], existing systems cannot overcome these problems.

Current systems fail to fully exploit the syntactical features of musical notation *i.e.* the strong syntax and knowledge about structure of written music. However, we have developed a recognition method which takes into account the constraints on music notation and actually uses them to improve the recognition process.

The method we shall present in this paper uses this established musical knowledge to control the entire recognition process, in particular the segmentation and labelling of objects. Fujinaga [7] states that music notation can be formalized using a grammar; therefore we propose the use of such a grammar to control the recognition process. The grammar adds another dimension, the context, to the process which allows us to find the correct segmentation for the object which is being matched to the rule in question.

In bottom up recognition systems, primitives are usually labeled as they are extracted using very local information, and the labels remain unaltered during the recognition. It is thus more probable that errors will be made that cannot be corrected. The method we propose means that primitives are only given a definitive label when once they are found to correspond to a certain rule, therefore labeling is carried out with a large amount of contextual information.

However, this grammatically formalized knowledge is only a score description. As we have already stated,

*In Proc. of the Third International Conference on Document Analysis and Recognition, Montréal, Canada, August 1995.

this description is not always capable of recognizing objects which pose segmentation problems (*e.g.* an accidental which touches a beam). Therefore, in order to be able to segment some touching objects and at the same time maintain the unity of knowledge, we propose a method of adapting the object description automatically so that touching objects can be recognized.

This method can be applied to different fields which have a strong syntax such as mathematical expressions, technical drawings, etc. The grammar we propose for musical notation is general enough to deal with full scores and has been validated by a musician.

In this paper we shall first discuss existing techniques, and then we shall go on to define our aims and point out the difficulties involved. We shall then give a brief description of the grammar and the associated parser. Next, we shall present an operator which is designed to modify the way the grammar parses the image to enable the system to deal with certain touching objects. Finally we shall give some results.

2 Related Work

In order to recognize musical scores, it is necessary to detect the staff lines (otherwise objects which are disconnected logically would appear as one single object). These can then be erased so that an initial segmentation process can be carried out on the objects (a technique used by most authors). The staff lines can be detected using different techniques, all of which treat the staff line as a real line. Only Carter [4] proposes a line method using a line-adjacency graph capable of dealing with staff lines that are not perfectly rectilinear (a relatively frequent case).

Once the objects have been segmented by erasing the staff lines, they are classified using different techniques (some authors use more than one): they are classified according to the bounding box size [10]; horizontal and vertical projections [7]; extraction of primitives such as notehead, stem, etc., using erosion, thinning or other techniques. Generally the label given to these primitives is not changed during the recognition process except in [8].

The main problems which then remain unresolved are [2, 4]:

- the scaling up of a prototype. Most of the systems which have been developed to date are prototypes adapted to simple scores. It is difficult to scale prototypes up to complex score systems (the same techniques cannot be used);
- the reconstruction of broken objects (noise);
- objects which touch when they should not.

The problems remain unresolved partly because a priori knowledge on the musical notation is not used to a sufficient extent. Although all the current systems for optical music recognition use some musical rules, these are usually selectively incorporated into the recognition process, and are not really formalized. Certain solutions do use some rules formalized by a grammar, but mostly for limited purposes (verification, error correction or final pitch calculation). They do not include graphical rules and are limited to simple scores.

Only Andronico [1] proposes the use of a grammatical formalization, which includes the graphical level. However, the grammars are only adapted to simple and monophonic scores. As mentioned previously, Fujinaga [7] states that the grammar for music notation is context-free and LL(k). Hence, it is possible to define a grammar to formalize the music writing rules which govern complex scores.

3 Aims

Our objective is to design a system:

- capable of recognizing full scores (orchestra scores) where each instrument has its own staff, and the staves are vertically connected;
- able to deal with complex scores, usually polyphonic (different voices on a single staff);
- that is reliable enough to avoid the need for human verification (using the redundancies in the musical notation);
- that can be extended to handwritten scores (if they are not too badly written);
- that uses a vocabulary which can be extended relatively easily;
- capable of recognizing only "classical" music notation. The notation used by some composers in the 20^{th} century can differ considerably from one work to another.

4 Difficulties

Touching Objects The scores to be recognized are complex and therefore have a high density of symbols. As a result, connections are often found between musical objects that syntactically should not be touching, causing some segmentation problems (Fig. 1).

Broken Objects Scanning produces some noises. In addition, initial documents may be of poor quality, giving broken objects. Usually, the objects affected in this way are small (*e.g.* tuplets's figures). Staff lines must first be removed for the recognition process but this can also produce broken objects (this is the most frequent case).

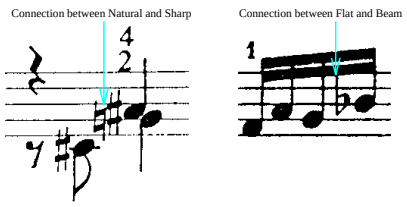

Figure 1: Examples of objects that should not touch

These segmentation problems are unavoidable. As traditional segmentation techniques at image level cannot solve the problems, we must rely as far as possible on the musical context.

5 Presentation of the Grammar

The objects on a score can be divided into two categories:

- constructs: (*e.g.* simple stemmed notes, beamed notes) composed of segments (such as stems, beams) and noteheads plus a set of construction rules which apply to these elements;
- symbolics: for instance clefs, accidentals, etc., these can be considered as characters. They can then be recognized using optical character recognition techniques (OCR).

Using this classification, grammar terminals are segments (part of a construct) and pixel arrays (for symbolics).

It is possible to distinguish between two levels of information on a musical score: a physical one corresponding to the way notes and their attributes (accidentals, accents, etc.) are formed are adjusted on the score; and a logical one corresponding to the syntactic way of using notes in written music, these notes are independent of the beaming, and have the attributes of an individual note: pitch, value, accidental, etc. This means that the grammar also has to be structured on two levels: a graphical one corresponding to the physical level, and a syntactic one corresponding to the logical level. As the grammar describes an image, it is necessary to introduce special operators which define the relative position of the elements.

- Position Operator:

  $$A\ \mathbf{P}\ B$$

  means A, and at the position **P** in relation to A, we find B.

- Factorization Notation (in association with the position operator):

  $$A\ (\mathbf{P1}\ B\ :\ \mathbf{P2}\ C)$$

  means A **P1** B and A **P2** C

- To define the extension of the rule for a single staff R to a rule for a system of staves :

  $$map_staff(R)$$

With this syntax, we can, for example, define a rule for a simplified beamed note:

beamedNote::= beam (**leftTip** noteGr :
[**closeV** [noteGr | consRest]]*:
rightTip noteGr)

We can also define the first rules for a simplified full score:

systOrchestra::= *map_staff*(heading) [systBar]+
heading ::= clef keySign timeSign
systBar ::= *map_staff*(barLine)
map_staff(barGr)
map_staff(barStep)

Where barGr represents the graphical part of the grammar in one bar of a single staff. At this level, only notes and their respective attributes are recognized (using rules like beamedNote). There is as yet no notion of voice (polyphony). The barStep rule represents the syntax for one bar of a single staff. At this level the results of the graphical part and a notion of Step (vertical cutting of the full score with a duration of the smallest duration of vertically aligned notes) are used to split notes into voices and to recognize voice or staff attributes, such as dynamic markings, phrasing slurs, etc. These two rules which are applied to a single staff are then automatically extended to cover a full score by the operator *map_staff*.

This grammar, described in more detail in [6] improves the recognition process (on segmentation problems) and is capable of detecting errors thereby achieving a greater degree of reliability.

6 Presentation of the Parser

The parsed structure that is entered is constructed at the image processing level. It is organized bar by bar and contains segments and connected components. As the image (and the grammar) has two dimensions, a cursor with the following components is incorporated into the parsing process: the current position in the image; the element to be parsed (a segment or a connected component) and a research zone, modified by the position operators.

In traditional parsers, this cursor corresponds to the head of the parsing list. The next element to parse is not necessarily the next one in the structure. It depends on the current position in the image and on the research zone.

The parsed structure is composed of segments (unlabeled) and of connected components, which do not necessarily each represent a musical object. For instance, a segment which has the features of a stem could also be part of a sharp. The parser can only give a definitive label to the segment (a Stem) using contextual information provided by the grammar.

Similarly, a connected component is not always segmented correctly to represent a symbol. Contextual information is again necessary to detect the presence of a symbol so that the component can be segmented accordingly. The parser can also cluster some unlabeled connected components, using the context to rebuild a symbol. The parser is automatically produced through a compilation of the grammar [5].

7 Specific problems where symbols touch constructs

Symbols can touch a construct either at segment level (stem or beam) or at head level (notehead or rest-head)(Fig. 2). In order to recognize these symbols, the system first needs to segment them correctly. This can be done by identifying the segments and heads of the constructs and removing them from the image so that the symbols are correctly ("naturally") extracted.

This method of segmentation is possible because the detector (using Kalman filtering [9]) can extract segments even if they are connected to other objects. The segment information is local enough for recognition to occur at a low level even when the object has been incorrectly segmented. But in the case of a construct, the description (*e.g.* the grammatical rule beamedNote) must recognize symbols (*e.g.* accidentals) as well as the segments. In order to recognize the beam of a beamedNote, it is first necessary to recognize all the noteGr, but for each noteGr, it is necessary to recognize the stem (a segment), the head and the possible accidental (a symbol). However, if the accidental touches the beam, it cannot be separated by removing recognized segments because the beam is still unidentified, as all the noteGr have not yet been recognized. This creates a vicious circle between the description of an object and the method used for its recognition.

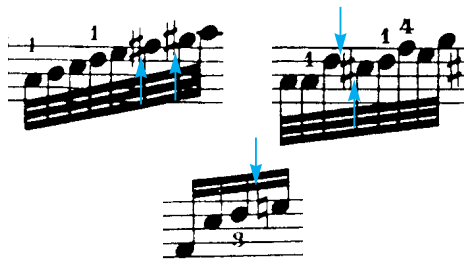

Figure 2: Examples of symbols touching constructs

One solution would be to separate the description of the segment and head in the construct from the note's attributes. The recognition process would then involve applying the construct rules (for segments and heads) and then the rules for the attributes. However, it would be difficult to identify which attribute went with which note and more importantly, the knowledge unity would be lost. As a result, it would be impossible to separate the knowledge from the program. In order to create a scaleable system and to be able to change and adapt rules without difficulty, it is essential to be able to separate this knowledge. Clearly, the vicious circle generated by the discrepancy between the description of an object and the method used for its recognition prevents this separation and, as a result, other systems tend to use only a small amount of knowledge and this directly in the program (not separated from it).

To overcome this problem, we have proposed the use of a Delay operator. This method will radically change the way a rule parses an object without changing the actual description of the rule itself.

8 Delay Operator

To keep the structural definition of the grammar, we have incorporated the Delay operator to modify the operational part of the parser.

For example, the noteGrU rule (graphical note with an upwards-oriented stem):

noteGrU ::= stem **closeLftDownTip** noteHead

will be given a final code containing the following operators (to make the syntax computable): `'::-'` the constructor of a rule; `'&&'` the connector of conjunction (the concatenation in the original grammar) and `'?$'` the position operator (**P**) in:

```
noteGrU BeamSeg ::-
  stem BeamSeg StemSeg &&
  ?$(closeLftDownTip StemSeg) && checkHead &&
  delay noteHead.
```

The operator Delay will systematically stop the clauses `noteGr` at the level of the predicate `noteHead`, until all of the bar has been parsed. The system must wait until the end of the bar has been reached because it is possible that a symbol may touch a segment that is part of a different construct than the one to which the symbol itself is attributed. To avoid combinatorial explosion, the conditions for the application of the rule `noteGr` have to be present before Delay can be used. For a note, this is the presence of a stem and a head. Therefore, the clause `noteGrU` succeeds if `checkHead` finds a notehead at the position defined by the position operator.

Delay then simply adds in a `noteGrU` parameter (a list of all the `noteHead` that have already been blocked) the predicate noteHead, its attributes and the current cursor. At the end of the bar, the heads and the segments recognized as stems or beams are deleted from the image, so that the touching symbols and constructs are segmented naturally.

Then the connected components are arranged into a new bar structure which gives the correct segmentation of the symbols. The delayed predicates can be awoken and the symbols (now correctly segmented) are recognized using OCR techniques (Fig. 3).

With the help of the saved cursor (for each note), all the `noteHead` rules will have the ability to link the correct attribute (accidentals, etc.) to the correct head. Note that this Delay mechanism is not the same as the freeze of Prolog II. Here, the delay is explicitly handled by the interpreter of the parser and does not depend on variable bindings. However, because goals are manipulated, this mechanism can be seen as a higher-order one.

9 Results

At present, a grammar has been defined for full scores, with different voices on a single staff, chords on a voice, accents, tuplets (*e.g.* triplet), pause, octave, dynamic markings, phrasing slurs, rhythmic slurs and

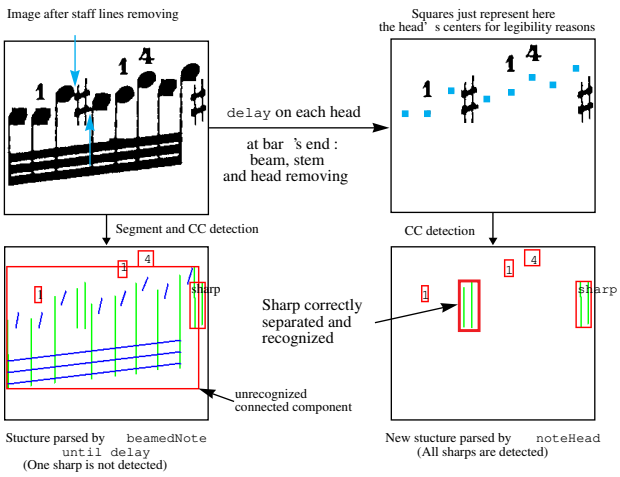

Figure 3: Structure before and after Delay

appoggiatura. Abbreviations, ornaments (except appoggiatura) and lyrics are not yet included. This work has been validated by a professional musician.

Most of the tools needed by the parser and some of the position operators have already been defined. As the grammar and the parser are mutually independent, it is possible to test the parser using a small grammar without losing its generality. We have defined a subset of the initial grammar to recognize full polyphonic scores with clefs, key and time signature, half, quarter, single eighth notes and beamed notes, accidentals, dots after a note, rests, bar lines, with the Delay operator and pitch processing. As the system can recognize full scores, it is also able to carry out two kinds of check: first on the number of beats in a bar (per voice) according to the time signature and second on the vertical alignment of notes in a system. These two checks enable the system to detect most errors and usually propose a method of correction.

The parser [5] is implemented in a compiled λProlog [3], a typed dialect Prolog integrating high-order unification. The choice of this language allowed us to produce an efficient prototype adapted to the complexity of the problem. The high-order level of λProlog allowed us to implement the Delay operator. Furthermore we have already written a grammar compiler which automatically produces the parser. At the end of the parsing, this system can point out unrecognized objects or bars which do not match the music syntax (which are likely to be poorly recognized).

As a result of the constraints on development time, we still have to incorporate the classification system for the symbols, and the segmentation and merging of

connected components. The current classification system is very simple, but we are currently carrying out improvements. These elements are tools for the parser, and should not be a problem for scaling the system. The parser is actually the kernel of the system and is independent of the grammar and the classification system, and of the segmentation and merging tools. Once the parser works, it will be possible to scale up the system by simply changing the grammar given to the parser, while the tools will be defined separately.

10 Conclusion

In this paper we have presented a recognition system for music scores that is fully controlled by a grammar. The grammar which currently exists can cope with full scores, with different voices on a single staff, with chords on a voice, and is also capable of recognizing accents, phrasing slurs, dynamic markings, etc. Unlike other systems, our approach formalizes all the rules used for recognition and permits maximum integration of the context. The formalization also produces a system which is homogeneous.

In most other systems, the syntax is used merely to check a label. Here, on the other hand, the syntax controls the entire recognition process, and therefore produces a more reliable label. Usually, the grammatical methods are used only at a high level, whereas this system goes back to the image level to produce an accurate segmentation and thus accurate recognition.

One advantage of the formalization of the grammar is the separation between the operating part of the system and the definition of musical rules. This separation means that rules can be easily modified or the system can be adapted for another kind of structured document. Infact, a new recognition system, using the same parser, can be constructed simply by defining a new grammar. This separation is possible using the Delay operator which modifies the way a rule parses the image to enable the segmentation of symbols which touch constructs. This operator can be used on other structured documents where a priori knowledge unity is important. The unity can be retained with Delay even where a difference exists between the way an object is described and the way an object can be recognized (usually linked to segmentation problems).

Moreover, this system provides a solution to the problems which currently remain unresolved, as described by Blostein [2] and Carter [4]: that is the reconstruction of broken objects, and the segmentation of touching objects. It should also help to solve the problem of scaling up the system. It is already able to deal with full scores, polyphony (for the moment only 2 voices per staff). With the help of the number of beats in a bar (according to the time signature) and the vertical alignment of notes (in full scores) the system can detect and correct recognition errors on note duration producing more reliable results.

References

[1] A. Andronico and A. Ciampa. On automatic pattern recognition and acquisition of printed music. In *ICMC, International Computer Music Conference*, pages 245–278, Venice, Italy, 1982.

[2] D. Blostein and H. Baird. A critical survey of music image analysis. In Springer-Verlag, editor, *Structured Document Image Analysis*, pages 405–434, Eds. H.S. Baird, H. Bunke, K. Yamamoto, 1992.

[3] P. Brisset and O. Ridoux. *The Compilation of λProlog and its execution with MALI.* Technical Report 687, IRISA, 1992. ftp://ftp.irisa.fr/local/lande.

[4] N. P. Carter and R. A. Bacon. Automatic recognition of printed music. In Springer-Verlag, editor, *Structured Document Image Analysis*, pages 456–465, Eds. H.S. Baird, H. Bunke, K. Yamamoto, 1992.

[5] B. Couasnon, P. Brisset, and I. Stephan. Using logic programming languages for optical music recognition. In *International Conference on the Practical Application of Prolog*, pages 115–134, Paris, France, April 1995.

[6] B. Couasnon and J. Camillerapp. Using grammars to segment and recognize music scores. In *International Association for Pattern Recognition Workshop on Document Analysis Systems*, pages 15–27, Kaiserslautern, Germany, October 1994.

[7] I. Fujinaga. *Optical Music Recognition using projections.* Master's thesis, McGill University, Faculty of Music, Montreal, Canada, 1988.

[8] H. Kato and S. Inokuchi. A recognition system for printed piano music using musical knowledge and constraints. In Springer-Verlag, editor, *Structured Document Image Analysis*, pages 435–455, Eds. H.S. Baird, H. Bunke, K. Yamamoto, 1992.

[9] V. Poulain d'Andecy, J. Camillerapp, and I. Leplumey. Kalman filtering for segment detection: application to music scores analysis. In *ICPR, 12th International Conference on Pattern Recognition (IAPR)*, pages 301–305, Jerusalem, Israel, October 1994.

[10] D. S. Prerau. Do-re-mi: a program that recognizes music notation. *Computer and the Humanites*, 9(1):25–29, January 1975.