A Structural Analysis Tool for Expressive Performance

Yuichi Uwabu (1) Haruhiro Katayose (2)(3) Seiji Inokuchi (1)(3)
(1) Graduate School of Engineering Science, Osaka University
uwabu@inolab.sys.es.osaka-u.ac.jp
http://www-inolab.sys.es.osaka-u.ac.jp/~uwabu
(2) Faculty of Systems Engineering, Wakayama University
(3) L.I.S.T

Abstract

This article describes an analytic framework that resolves the structural ambiguity of GTTM. First, we describe a method to represent the musical surface and structure by the functional language ML grammar. Second, we describe an improvement of the analytical strategy. This strategy resolves ambiguity by checking a priority. The priority defined the frequency of co-occurrence relations between two events. The co-occurrence relations are extracted from corpora. The analytical model is constructed based on that representation and strategy.

1 Introduction

Human being can appreciate music and generate expressive performances. Investigating such human musical competence is a main subject of music research. In our research, we focus on interpretation process that connects musical surface described in scores and expressive performances. Our definitive goal is to construct a musical interpretation model as a computational model.

Musical interpretation means that a performer extracts composer’s idea using his/her musical knowledge and experience and add proper expression to the piece. In the field of musicology and psychology, this kind of research becomes major and many theories and hypothesis have been proposed. Recently, some promising theories applicable to computational model has been proposed; Lerdahl and Jackendoff’s “A Generative Theory of Tonal Music (GTTM) [1]” and Narmour’s “Implication-Realization Model [2].” Several researchers are engaged in learning expression rules [3]. We have developed a system that extracts expression rules using structure in their theory and multiple regression analysis. Commonality of these approaches is learning relations between structure and expression in a given performance. However, when extracted rules are applied to a new piece, the system requires not only expression rules but also structural information of the given piece. Therefore, the structural analysis on computer system is indispensable.

We started with the implementation of GTTM, but there are some problems to be solved. First, we have to formalize the knowledge for representing musical surfaces and recursive hierarchical structures. That representation is related to the efficiency of the analytical strategy. Second, we have to decide a priority of applying preference rules when conflict occurs. However, the problem is that GTTM does not formalized how to apply rules. A knowledge of performer decides the priority. The concept like intra- and extraopus style in terms of Narmour’s theory is important. In terms of GTTM ‘parallelism’ is equivalent to that concept, but details are not discussed. We consider the parallelism to associate similarity from the knowledge. A frequent occurrence will have high priority. We pay attention to a co-occurrence relation between two notes.

In this article, we take these problem into account. In the following section, the representation and the strategy are described in detail.

2 Outline of GTTM

GTTM was developed as an approach to the analysis of tonal music. While some critical reviews have been written [4], it is regarded as one of the most
promising theories. GTTM has four components; grouping structure, metrical structure, time-span reduction and prolongational reduction. Each component is not independent, but related with each other. Figure 1 shows their relation.

![Diagram showing the relation among GTTM components]

We are going to apply GTTM’s structures to extracting expression rules using. The most important structure linked directly expressive performances is the prolongational reduction. That structure means how the motion among the pitch events of a piece tenses or relaxes, and inherits elements of other components. Therefore, we show how to induce the prolongational reduction as example.

### 3 Representation

This section describes how to represent the musical surface of tonal music and the prolongational tree. The musical surface can be classified into several ranks [5] [6]. The rank of note and chord is handled here. Each rank has many features, and their representation depends on musical styles. The features efficient in this analysis is used in the following definition. The musical surface is reduced to the recursive hierarchical structure. The prolongational tree is the result of the prolongational reduction. There are several branching patterns. The representation must be a reflection of these characteristics.

All representation is defined by the ML grammar. ML is the functional language like Lisp. We don’t go into details about its definition.

#### 3.1 Note

Notes are the most basic unit, and defined with two features: pitch and octave. The pitch specifies name of chromatic scale: c, f#, gb. The octave specifies octave of note.

```ml
datatype note = Note of {pitch:string, octave:int};
```

A simple example is shown below. The symbol “$$\Rightarrow$$” is ML prompt. The following is an equation that user define. Here, the “C4” is bound as a variable. The next line is a reply of ML.

*For example:*

```ml
- val C4 = Note {pitch = "c", octave = 4};
val C4 = Note {octave=4,pitch="c"} : note
```

#### 3.2 Chord

Chords are defined with four features: root, type, inversion and key. The root is a root note defined in section 3.1. The type (typ in the representation) and inversion particularizes a chord name which has a same root. The key specifies the underlying tonality.

```ml
datatype chord =
  Chord of {root:note,typ:string,
inversion:string,key:string};
```

An example shows a description of “I6” chord. A reply of ML includes the value of “C4” as a bound variable.

*For example:*

```ml
- val I6 = Chord {root=C4,typ="maj",
inversion="6",key="Cmaj"};
val I6 =
  Chord {inversion="6",key=\"Cmaj\",
  root=Note {octave=4,pitch=\"c\"},typ=\"maj\"} : chord
```

#### 3.3 Prolongational Tree

We define the prolongational tree which has a nature of a recursive hierarchical structure. The prolongational tree has three tensing patterns by right branching and three relaxing patterns by left branching (figure 2).

The ptree type has seven constructors. The long name like “right strong prolongation” are shortened as “RSP”. Each constructor is defined recursively as including the ptree type itself. The ptree list is a sequence of items of the ptree type, and the nil constructor is the list containing nothing. The first item of list has the highest prolongational importance.
datatype ptree =
    Head of (note*chord) * ptree list |
    RSP of (note*chord) * ptree list |
    RWP of (note*chord) * ptree list |
    RP of (note*chord) * ptree list |
    LSP of (note*chord) * ptree list |
    LWP of (note*chord) * ptree list |
    LP of (note*chord) * ptree list ;

An example shows the prolongational tree of figure 3 as the ptree type.

For example:

Head( (C5,I), [
    LWP( (E5,I), [
        RSP( (E5,I), nil )
    ] ),
    LP( (D5,V), [
        LP( (F5,IV), nil )
    ] )
]);

4 Analytical Strategy

In GTTM, the insufficiency of strictness in the preference rule causes the structural ambiguity. We define the ambiguity as a conflict situation in branching selections of any one note (Figure 4). It induces a lot of inadequate analysis results in local and global level and drops the efficiency of the analysis. Therefore, when the conflict occur, our analytical strategy resolves the ambiguity to induce one of the most prior results. In local level, a priority is defined by a frequency of the co-occurrence relation between two elements. The co-occurrence relations are extracted from corpora by evaluating similarities of types and features; the type of branching pattern and
selected rule and the feature of note and chord. The corpora contain the prolongational tree of the piece. The frequency of co-occurrence relations is accumulated as a co-occurrence knowledge. Figure 5 shows an overview of our system.

The details of the analytical strategy are described below. We define some equations to formulate a strategy, and regulate how to apply them.

Let the list of input data be the elements \([e_1, \ldots, e_n]\). \(f(e_i)\) is the priority of branching pattern between the element \(e_i\) and the other. It can be defined as follows:

\[
f(e_i) = \left\{ \begin{array}{ll}
\max_i \left\{ \frac{C(a_i)}{\sum_{x \in A} C(x)} \right\} & : \text{conflict} \\
1 & : \text{no conflict}
\end{array} \right.
\]

where \(A=\{a_1, a_2, \ldots\}\) is a set of assumptions \(a_i\) which are conflict situations of branching pattern and \(C(a_i)\) is the frequency of the co-occurrence relation in each assumption \(a_i\). The co-occurrence knowledge base has a sum of co-occurrence relations in a specific situation. The co-occurrence knowledge gives the value of the priority \(C(a_k)\).

\(S[i,j]\) is the likelihood of the prolongational tree which is the sum total of \(f(e_i)\). It is defined recursively as follows:

\[
S[i,j] = S[i,k] + S[k,j] + f(e_i) \quad (i + 1 \leq k \leq j - 1)
\]

\[
S[i,i+1] = 0
\]

When the time-span tree and the elements \([e_1, \ldots, e_n]\) as note and chord is given,

(a) select the highest time-span importance as a prolongational head,

(b) apply the preference rules and calculate the priority \(f(e_i)\) when a conflict occurs in the current situation,

(c) repeat the step (b) and calculate the maximum value of the \(S[1,n]\).

This strategy can handle the structural ambiguity quantitatively. Figure 6 shows a simple example. Each value is the likelihood of branching patterns. Each of the likelihood of tree is 1.9 (figure 6a) and 2.1 (figure 6b). Our strategy prefers the result in figure 6b.

5 Conclusions

This article has presented the representation of musical surface and structure and the strategy which resolves the structural ambiguity with the priority. The representation offers sufficient features to evaluate co-occurrence relations. The analytical strategy shows the validity of the co-occurrence knowledge. The remained problems are the evaluation method of co-occurrence and a shortage of corpora. It causes a shortage of the co-occurrence knowledge, which influence the efficiency of the analytical strategy. In future work, we extract the co-occurrence knowledge at more abstract level from sufficient corpora. The extracted knowledge will apply the statistical analysis to verify the adequacy of the preference rule.

References