属性文法による楽譜の定式化と自動認識に関する研究

5S－9

日和崎 祐介　原田賢一
慶應義塾大学理工学部

1　はじめに

音楽は、計算機科学の芸術への応用として古くから研究がなされている分野であり、現在では計算機を利用した音楽処理は実用的な段階にまで達している。従来、楽譜を計算機で処理可能な形で蓄える音楽データベースの構築を目的とする楽譜の自動認識系の研究がなされてきた。しかし、これらは単なる認識科学の応用として低レベルの記号認識を行い、音符の意味などはアドホックな手法を用いて評価していることが多い。

本稿では、音楽の形式的記述である楽譜を、属性文法として定式化することを提案する。ここで、楽譜の論理構造には文脈自由文法記号、意味には意味規則を用いる。また、この定式化に基づいた楽譜自動認識の翻訳系 (高レベル認識部) への応用について述べる。

2　楽譜の定式化

音符や休符を基本記号と見なし、楽譜は整構造を持つそれらの列と考えると、形式言語として定式化することができる。この定式化を行うには、まずその文法クラスについて考察をしなければならない。音楽自体はチョムスキーの階層構造上では自然言語と同様に無制限な文法をもつ。しかし、楽譜は制限の厳しい形式的な記述方法で、その生成規則は楽譜記法として定められている。本研究では、楽譜を**属性文法**によって定式化を試みた。

2.1　楽譜の基底文法

楽譜上の記号は二次元に記述され、縦と横方向にそれぞれ音の高さと時間が表現される。文章と同様に左から右へと読み進み、右側の記号は左側の記号より一定時間経過したことを示す。また、五線を基準として、高い位置にある記号は下に位置する記号より周波数が高いことを示す。つまり、楽譜は音楽を時間と周波数方向と二次元に広がりを持つものとして形式化されている。

楽譜の記述規則は、プログラミング言語のように生成規則によって表現することができる。楽譜の記号の出現順は表記規則によってほぼ定められており、決められた順序の記号列には名称が付けられている。さらに、これらは図１のように、時間軸方向に論理的入れ子構造を持つ。したがって、これらは生成規則の非終端記号、そして楽譜上の記号は終端記号と対応づけることができる。

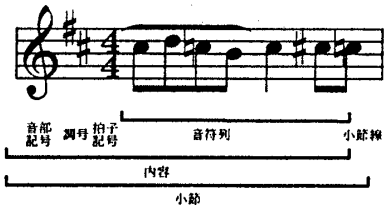

図 1: 記号の論理的構造の一部

ここで、論理的名称の入れ子の親は一つの名称しか持たない。したがって、この形式は文法の生成規則の左辺は必ず１つの非終端記号しか持たない**文脈自由文法**と同じ文法クラスで表すことができる。

2.2　楽譜の意味

次に、その意味を規定する方法を考察する。楽譜の主体は音符や休符などの事象を表す離散記号で、それらの意味付けを行うための属性付加を目的とした記号が周辺に印刷される。また、それらの意味づけを行うための環境を設定する環境記号がすべての楽譜の冒頭に記述される。これらの環境記号は途中で変更されることもある。したがって、音符の全ての属性は、前出もしくは同時に出現する記号によって値が定まる。なぜならば、楽譜が時間軸に広がりを持つからで、右側に出現する記号によって値が定まるようでは、動作を再生する際に (つまり演奏する際に) 非実用的である。属性は左から右へと受渡されるため、楽譜の意味は**L属性的**であることが直観的に理解できる。

Fahmy H. and Bolstein D. の研究 [1] は、グラフ文法を用いて属性記号と音符の結び付きを計算する手法があるが、その手法では計算時間と空間の面で効率が悪い。また、音楽を楽典的分析にもとづき解析を行う Lerdahl[3] の研究は解析中に複数の構文木を生成しさらにその構文木間につながりを持つため、非常に複雑なものとなっている。

以下に属性文法として定式化された楽譜の簡略化した例を示す。ここでは、生成規則に、調性を表す属性

A formalization of musical score using attribute grammar and its application to automatic score recognition
Yuusuke HIWASAKI, Ken'ichi HARADA
Department of Computer Science, Keio University
3-14-1 Hiyoshi, Kouhoku-ku, Yokohama 223, JAPAN

key の意味規則が { } で囲まれて付随する。小節 (bar) 内の音符 (note) の調性 (*key*) が調号 (key) と臨時記号 (accidental) によって定まる規則を示している。

```
bar        →  content barline
              { bar.key      = content.key; }
content    →  clef key note_list
              { content.key = key.key;
                note_list.key= key.key; }
note_list  →  note_list accidental note
              { note_list(1).key
                  = accidental ⋃ note_list(2).key;
                note.key
                  = accidental ⋃ note_list(2).key; }
```

3 楽譜の自動認識への応用

本研究では、属性文法による楽譜記述の応用、そしてその検証として楽譜の自動認識における楽譜データ翻訳器の実装を行った。

3.1 従来の研究との比較

認識科学の応用の対象として、従来楽譜認識は古くから研究がなされているが、記号を認識することに重点がおかれ、記号の属性を決定はアドホックに処理をしており、記号の出現順や相対的な位置関係を形式化する試みはない。また、これらの研究では認識された楽譜を計算機上でどのように表現するかについての考察が不十分であり、認識を専門としているため、楽譜の楽典的分析などへの応用も難しい。

3.2 自動認識系への実装

処理系は、まず楽譜を画像として取り込む入力部分、楽譜記号の認識を行ない記号の位置表現を生成する認識部と、その位置表現より最終的により一般的な計算機上の音楽表現に変換する翻訳部の三つに分けられる。また、認識部の出力については、翻訳部で構文と意味規則の二つの側面から検査を行うことができる。この処理系の全体の流れは図2のとおりである。

この翻訳部は先に述べた楽譜の属性文法を属性文法評価器生成系 Rie[4] と C 言語を使用して記述し、コード生成ルーチンを意味規則に加えることによって実装を行った。この評価器への入力としては、記号を出現順に配置したものを与える。

処理系の出力データは、その用途によって必要とされる形式が異なるため、選択には注意が必要である。この出力形式としては、MIDI や現在 ISO で規格化されつつある SMDL [5] などが挙げられる。ここで、構文主導翻訳にもとづいて出力形式ごとにコード生成のバックエンドを記述することだけでさまざまな出力形式に対応することが可能である。今回の実装では、出力コードとして MIDI コードを選択し、演奏可能なマルチトラックの標準 MIDI ファイル形式を得ることができた。

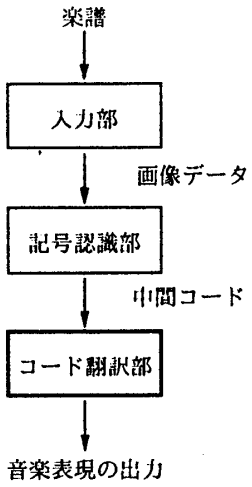

図 2: 楽譜自動認識系の流れ

4 まとめ

本稿では、楽譜の属性文法による定式化と、その応用としての楽譜の高レベル自動認識系への実装について述べた。構文主導翻訳を用いて、コード生成のルーチンを出力形式ごとに記述するだけで様々な出力コードに対応できる柔軟な設計ができた。同様の手法を用いて、目的に即した意味規則を拡張すれば、楽典的な解析などを行う処理系にも応用できると考えている。このような楽譜の定式化は、MIDI や SMDL などの汎用的なデータ生成の分野だけでなく、計算機で音楽を扱うための手法として有用なことと考える。

参考文献

[1] Fahmy H. and Bolstein D.: "A Graph Grammar for High-level Recognition of Music Notation," ICDAR (1991) pp.70–78.

[2] Lerdahl.F. and Jackendoff.R.: "An Overview of Hierarchical Structure in Music," Music Perception. Berkeley CA. University of Californai Press.

[3] Sassa M., et al: "Rie - Introduction and User's Manual version 1.0.4," Tokyo Institute of Technology (1994).

[4] ISO/IEC: "Information Technology - Standard Music Description Language," Committee Draft. ISO/IEC CD 10743 (1991-03).