1S-8

関数型言語を用いた宣言的な楽曲分析システム

小長谷 康治† 山本 晋一郎‡ 大久保 弘崇‡ 粕谷 英人‡

†愛知県立大学大学院情報科学研究科 ‡愛知県立大学情報科学部

1 はじめに

1.1 背景

コードスケールシステムはポピュラー音楽における楽曲分析の基礎であり，コードに対して使用できるスケールを対応させる手法である．作曲・編曲・アドリブなどの支援を計算機で行う場合において，コードスケールシステムは重要である．コードスケールシステムに基づいた楽曲分析システムは存在しているが，より高い拡張性・保守性のあるものが求められている．

1.2 目的

本研究では関数型言語を用いて宣言的な楽曲分析システムを実装することを考える．ここでいう分析とは，楽曲の調，コード進行，メロディから各コードに対応するスケールを求めることをいう．実装には関数型言語 Haskell を用い，コード，スケールや音程などの分析に必要な音楽理論を利用しやすい形で実装し，高い拡張性や保守性を目指す．分析システムの対象は，今日のポピュラー音楽で広く用いられている，コードスケールシステムに基づいて作曲された楽曲とする．

2 音楽理論のコード化

2.1 Haskore

Haskore[1] は Haskell で記述された音楽記述言語で，Haskell を使った音楽情報処理研究で多く利用されており [2]，本研究でも Haskore を利用する．しかし Haskore は楽曲を生成・変換することを第一目的に設計されていて，分析に必要なコードやスケールをそのままでは扱えないため，それらを新たに定義する必要がある．

2.2 Scale データ型

分析に必要な音楽理論のひとつとして，新たに定義したスケールを表すデータ型を以下に示す．

リスト 1: Scale データ型

```
1  data Scale = Scale {
2    rootNoteOfScale :: PitchClass,
3    scaleName :: ScaleName,
4    scaleNote :: [ScaleNote]
5  }
6
7  data ScaleNote = ScaleNote {
8    degreeOfScaleNote :: Interval,
9    kindOfScaleNote :: KindOfScaleNote
10 }
```

他の音楽情報処理研究で見られる，演奏のみを目的としたスケールの実装では，単にスケールの構成音のみを定義する．本システムではスケールの構成音以外にも構成音の度数や種類も扱えるデータ型として定義し，一般的な楽曲分析に広く使われることを考慮した．またその他にもコードを表す HarmonyChord データ型や調を表す Key データ型など，楽曲分析に必要な 10 個のデータ型を汎用性を考慮して定義した．

3 コードパターンによる楽曲分析

3.1 コードスケールシステム

コードスケールシステムでは，調とコードからそのコードに使用できるスケールの候補が決まる．ただし実際の楽曲では調が一時的に変化する部分転調が起こりうるため，楽譜に書かれている調とコードだけからスケールの候補を決定できない．そこで，コードを複数のまとまりで扱うコードパターンを考える．部分転調はコードの前後関係で推測できる．

3.2 コードパターン

文献 [3] に基づいて 45 種類のコードパターンを決定した．コードパターンの一部を表 1 に示す．

表 1: コードパターン抜粋

パターン名	コードパターン	機能	調
トゥー・ファイブ	IIm - V	⟨SD⟩ - D	I Major
偽終止	IVm - IIIm	SDM - ⟨T⟩	I Minor

楽曲のコード進行に対してこれらのコードパターンをパターンマッチングすることにより，楽曲分析を行

Music Analysis System using Functional Language

†Koji Obase, Graduate School of Information Science and Technology, Aichi Prefectural University.

‡Shinichiro Yamamoto, Hirotaka Ohkubo and Hideto Kasuya, School of Information Science and Technology, Aichi Prefectural University.

う．また同じ機能を持つコードをコードグループ，スケールの候補をスケールグループにまとめ，コードパターンとは別に定義してある．コードグループ，スケールグループの一部を表 2 に示す．

表 2: コードグループ，スケールグループ抜粋

グループ	コード	スケール
IIm	IIm, IIm7	Dorian
IVm	IVm, IVm7	Dorian, MelodicMinorUp

このようにコードパターンとコードグループ，スケールグループとを分け，知識を階層的に実装することで，拡張性・保守性を高めている．またこのように宣言的なパターンを記述することに対して，関数型言語の特徴である代数的データ型やリストが役に立っている．

3.3 分析手順

分析手順は，入力楽曲のコード進行に対してコードパターンのマッチングを行い，それぞれのコードに対するスケールの候補を決定する．そのあと実際のメロディと比較し，正しいスケールを決定する．

3.4 実装

楽曲分析システムの概要図を図 1 に示す．入力ファイルには MusicXML 形式を使用する．楽曲の調，コード進行，メロディを入力とし，分析結果としてスケールやケーデンス等を出力する．実装したシステムは大きく分けて以下の 3 つに分類できる．

分析エンジン 楽曲分析を行うエンジン．大きく，コードパターン検出モジュールとスケール選択モジュールの二つがある．約 170 行．

知識 コードやスケールの種類とその構成音や分析に必要なコードパターン．約 370 行．

その他 分析結果をプリティプリントする関数や MusicXML パーサなど．約 400 行．

分析エンジンと知識を分離し，分析エンジンを修正しなくても新たに知識にパターンを追加すれば分析をカスタマイズできる設計になっている．

4 評価

文献 [4] で実例として紹介されている 32 小節からなる長調の楽曲を実装したシステムを用いて楽曲分析し，予備的な評価を行った．この楽曲にはジャズで頻出する部分転調や代理コードが多く使われているので，分析システムの評価に適している．

分析結果の正誤判断を表 3 に示す．分析結果が文献 [4] の解説と一致した場合を正解とする．文献の解説と一致しない場合で，分析結果を別の正解としてみなせる場合を別解，それ以外を不正解とする．提案手法では分析できない場合を分析不能とする．

表 3: 分析結果の正誤判断

分析スケール数	正解	別解	不正解	分析不能
36	30	0	0	6

全体の 83% についてスケールを正しく分析することができた．分析不能に分類されたものの理由は，本システムでは考慮していないドミナントアプローチによる特殊なスケール外音が使用されており，スケールを正しく決定できないためであった．これらを検出するための新たなコードパターンを追加すれば，分析可能であると考えている．

5 まとめと今後の課題

関数型言語 Haskell を用いて楽曲分析に必要な音楽理論を利用しやすい形で定義し，実際にそれらを用いて楽曲分析システムを実装した．分析を行うエンジンと分析に必要な知識を分離し，分析システムの拡張性・保守性を高めた．

今後の課題として，本システムでは考慮していなかった特殊なスケール外音に対応したコードパターンを考案する必要がある．また評価実験が不十分であると考えられるため，様々な楽曲で評価を行う．

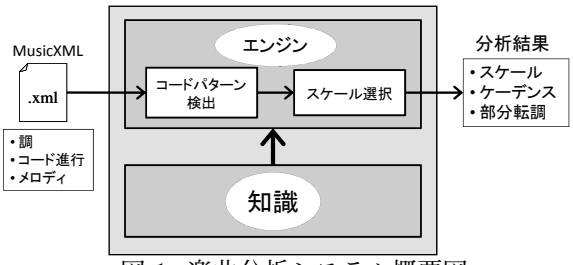

図 1: 楽曲分析システム概要図

参考文献

[1] Haskore
http://www.haskell.org/haskellwiki/Haskore .

[2] HasChorus
http://meltin.net/hacks/haskell/ .

[3] 北川祐: “ポピュラー音楽理論”, リットーミュージック, 2004 .

[4] 谷口広志: “THE REAL JAZZ GUITAR”, ケイ・エム・ピー, 2004 .