Vol.2017-MUS-115 No.25
2017/6/17

プログラムからの楽曲作成

渡邉 彰吾[1,a)]　六沢 一昭[2,b)]

概要：本稿は,Perl プログラムから楽曲を作成するシステムについて述べる. プログラムを見た時に感じる印象を, 聴いて楽しむことのできる楽曲という形で表現することがこのシステムの目標である. 楽曲は, プログラムの 1 行を 1 小節として作成する. まず, プログラムを見た時の印象に影響する情報を解析し, 各小節のコードを決定する. そして, コードをもとにメロディとリズム, それにハーモニーを作成する.

1. はじめに

本稿は, プログラムから自然に楽しめる楽曲を作成するシステムについて述べる. プログラムを見た時に感じる印象を, 聴いて楽しむことのできる楽曲という形で表現することがこのシステムの目標である.

プログラミング学習支援やプログラム開発支援などを目的としたソフトウェア可聴化システムがこれまで多く開発されてきた [1][2][3][4][5][6]. これらの目的は主としてプログラムの実行状況の可聴化であった. また, 主として生成されるものは, 楽曲というよりは音の集合であり, 聴いて楽しむというエンターテインメント性を重視したものはほとんどなかった.

本システムは, Perl 言語で記述されたプログラムを対象とする. ネスト構造や制御構造などといった, プログラムを見た時の印象に影響する情報を解析し, 楽曲を生成する.

2. 自然な楽曲の作成

自然な楽曲を作成するために着目したコードと規則を以下に示す.

2.1 ダイアトニックコード (Diatonic Chord)

ダイアトニックコードとは, スケール上の音で構成される三和音 (Triad) または四和音 (Tetrad) のことである. またスケールとは音階のことで, オークターブ内の音の配列である. 本研究では, 三和音と長調 (メジャースケール) を扱い, 使用するコードの一覧を表 1 に示す.

表 1　各調のダイアトニックコード (三和音)

調＼度数表記	I	IIm	$IIIm$	IV	V	VIm	$VIIm^{\flat 5}$
ハ長調	C	Dm	Em	F	G	Am	$Bm^{\flat 5}$
嬰ハ長調	$C\#$	$D\#m$	Fm	$F\#$	$G\#$	$A\#m$	$Cm^{\flat 5}$
ニ長調	D	Em	$F\#m$	G	A	Bm	$C\#m^{\flat 5}$
嬰ニ長調	$D\#$	Fm	Gm	$G\#$	$A\#$	Cm	$Dm^{\flat 5}$
ホ長調	E	$F\#m$	$G\#m$	A	B	$C\#m$	$D\#m^{\flat 5}$
ヘ長調	F	Gm	Am	$A\#$	C	Dm	$Em^{\flat 5}$
嬰ヘ長調	$F\#$	$G\#m$	$A\#m$	B	$C\#$	$D\#m$	$Fm^{\flat 5}$
ト長調	G	Am	Bm	C	D	Em	$F\#m^{\flat 5}$
嬰ト長調	$G\#$	$A\#m$	Cm	$C\#$	$D\#$	Fm	$Gm^{\flat 5}$
イ長調	A	Bm	$C\#m$	D	E	$F\#m$	$G\#m^{\flat 5}$
嬰イ長調	$A\#$	Cm	Dm	$D\#$	F	Gm	$Am^{\flat 5}$
ロ長調	B	$C\#m$	$D\#m$	E	$F\#$	$G\#m$	$A\#m^{\flat 5}$

2.2 カデンツ (Kadenz)

カデンツとは, 終止形和音進行のことである. 文章で言う所の起承転結のようなものである. コード進行はカデンツに則って構成する. これにより, 自然な楽曲の作成を試みる. カデンツを組み立てる上での和音機能は 3 種類存在する.

トニック **(T)**　– 最も安定した響きを持ち, 曲のはじめと終わりによく使われる.

ドミナント **(D)**　– 不安定な響きを持ち, トニックへ戻ろうとする.

サブドミナント **(S)**　– やや不安定な響きを持つ.

また, カデンツには以下の 3 種類の型があり, この関係を図 1 に示す.

カデンツ第 **1** 型 **(K1)**　– T-S-T

カデンツ第 **2** 型 **(K2)**　– T-S-D-T

カデンツ第 **3** 型 **(K3)**　– T-D-T

1　千葉工業大学 情報科学研究科 情報科学専攻
2　千葉工業大学 情報科学部 情報工学科
a)　watanabe.syogo@rok.cs.it-chiba.ac.jp
b)　rokusawa.kazuaki@it-chiba.ac.jp

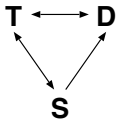

図 1　カデンツ

各 3 和音の機能別分類を表 2 に示す.

表 2　3 和音の機能別分類

和音機能	該当する 3 和音
トニック (T)	I,$III m$,$VI m$
サブドミナント (S)	$II m$,IV
ドミナント (T)	V,$VII m^{\flat 5}$

3. プログラムから得られる印象

プログラムを見た時, 以下から印象が作られるのは自然であると思われる.

3.1 ネストの変化

ネストは分岐やループがあると変化する. プログラムのネストが一定の部分からは単調な印象を受ける. これに対して, ネストが深くなる/戻る部分からは動きを感じる.

3.2 制御構造の種類

while, for, if (then, else) といった分岐やループの制御構造を見ると, それぞれに応じたプログラムの振る舞いを予測する.

3.3 各行の複雑さ

閉じ括弧のみの行や, 変数に定数を代入するだけの行は単純に感じる. 一方, 多くの変数や演算子, 関数呼び出しを含んだ計算式や条件式などからは複雑な印象を受ける.

4. プログラムと楽曲の対応

音楽の雰囲気を決める「コード進行」と音楽の三要素「リズム, メロディ, ハーモニー」に着目して楽曲を作成する. また, プログラムの 1 行から楽曲の 1 小節を作成する. 主に, 3 で述べた項目と楽曲の対応を以下に示す.

4.1 コード進行

ネストは一つずつしか変化せず, ネストが奥深くまで行くほどプログラムが進行していると考えた. これより, ネストの深さをコード進行で表現する.

ネストの深さに応じて適当にコードを割り当てるのではなく, カデンツに則ってコードを決定する. これにより, 自然な楽曲の作成ができると考えられる. ネストの深さに応じて各小節のコードを決定する. ハ長調の場合の対応を表 3 に示す. また, コードの遷移例を図 2, 図 3 に示す.

表 3　ネストの深さとコード (ハ長調)

ネストの深さ	0	1	2	3	4	5	6	7
コード	C	G	Am	Dm	Em	F	C	$Bm^{\flat 5}$
和音機能	T	D	T	S	T	S	T	D

ネスト	コード	
0	C(T)	`for ( $i = 0 ; $i <= 0 ; $i++ ) {`
1	G(D)	`  if ( $i % 2 == 0 ) {`
2	Am(T)	`  print "$i";`
1	G(D)	`  }`
0	C(T)	`}`

図 2　コードの遷移例 1

ネスト	コード	
0	C(T)	`if ( $i % 2 == 0 ) {`
1	G(D)	`  print "#";`
0	C(T)	`}`
0	C(T)	`else {`
1	G(D)	`  print "*";`
0	C(T)	`}`

図 3　コードの遷移例 2

表 3 のように対応させることにより, ネストが図 2 や図 3 のように変化してもカデンツの型が保たれていることがわかる.

4.2 リズム

プログラム 1 行のパーツ (構成要素) 数はプログラムの複雑さを表していると考えた. これより, プログラムの複雑さはリズムの細かさで表現する.

1 行をパーツに分解し, 分解したパーツ数に応じて 1 小節の音符数を決定する. パーツ数とリズムの対応を表 4 に示す. また, パーツの分解例を図 4 に示す.

表 4　パーツ数とリズム

パーツ	リズム	パーツ	リズム
1	𝅝	9	♩ ♫ ♫ 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯
2	𝅗𝅥 𝅗𝅥	10	♫ ♫ ♫ 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯
3	𝅗𝅥 ♩ ♩	11	♩ ♫ 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯
4	♩ ♩ ♩ ♩	12	♫ ♫ 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯
5	♩ ♩ ♩ ♫	13	♩ 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯
6	♩ ♩ ♫ ♫	14	♫ 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯
7	♩ ♫ ♫ ♫	15	𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯
8	♫ ♫ ♫ ♫	16	𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯 𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯𝅘𝅥𝅯

```
for ( $i = 0 ; $i < 9 ; $i ++ ) {
    if ( $i % 2 == 0 ) {
        print "$i¥n" ;
    }
}
```

図 4　パーツ分解例

4.3　メロディ

楽譜での音符は文章での文字と言えると考えた. これより, プログラムの文字をメロディで表現する.

表 5 にハ長調で使用するコードの構成音を示す. また, 構成音の一覧を図 5 に示す. メロディ値とは, パーツを構成している文字コードの合計値を 7 で割った余りである.

メロディ値 7,8 はメロディの和音専用の音である. メロディの和音については 4.4.1 で説明する.

表 5　コードの構成音 (ハ長調)

コード ＼ メロディ値	0	1	2	3	4	5	6	7	8
C	$C4$	$E4$	$G4$	$C5$	$E5$	$G5$	$C6$	$E6$	$G6$
Dm	$D4$	$F4$	$A4$	$D5$	$F5$	$A5$	$D6$	$F6$	$A6$
Em	$E4$	$G4$	$B4$	$E5$	$G5$	$B5$	$E6$	$G6$	$B6$
F	$F4$	$A4$	$C5$	$F5$	$A5$	$C6$	$F6$	$A6$	$C7$
G	$G3$	$B3$	$D4$	$G4$	$B4$	$D5$	$G5$	$B5$	$D6$
Am	$A3$	$C4$	$E4$	$A4$	$C5$	$E5$	$A5$	$C6$	$E6$
$Bm^{\flat 5}$	$B3$	$D4$	$F4$	$B4$	$D5$	$F5$	$B5$	$D6$	$F6$

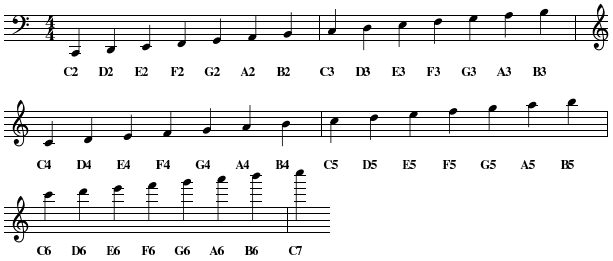

図 5　構成音

4.4　ハーモニー

メロディと伴奏の和音の対応を以下に示す.

4.4.1　メロディの和音

複数の要素を持つ配列やハッシュをメロディの和音で表す. 配列は 2 つの和音で, ハッシュは 3 つの和音で表す.

4.4.2　伴奏

メロディを支え引き立てる伴奏は主要部分を強調することに適していると考えた. これより, プログラムの骨格となる制御構造を伴奏で表現する.

制御構造ごとの伴奏例を図 6 から図 9 に示す.

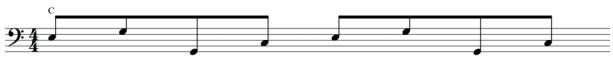

図 6　for の伴奏

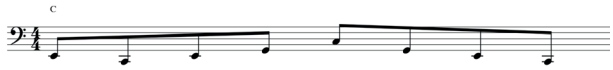

図 7　while の伴奏

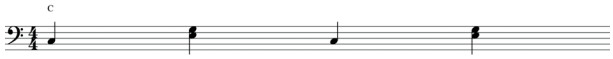

図 8　if の伴奏

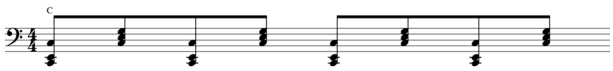

図 9　else の伴奏

4.5　転調

プログラムはサブルーチンが入ると処理が大きく変化すると考えた. これより, サブルーチンを転調で表現する.

5.　システムの構成

楽曲作成対象プログラムは Perltidy[*1] によって整形され, 本システムによって楽曲が作成される. 作成された楽曲は MML[*2] で記述される. システムの構成を図 10 に示す.

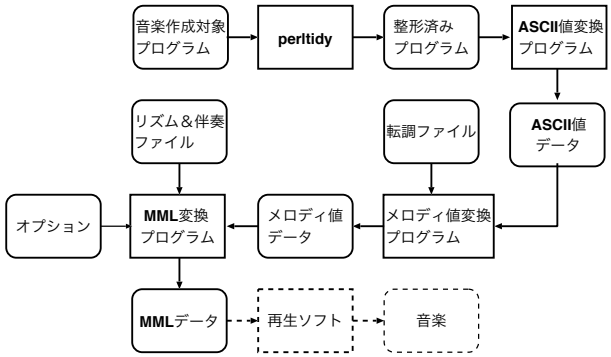

図 10　システム構成図

*1　Perl プログラムの整形ツール
*2　Music Macro Language, 楽曲記述言語の一つ

6. 作成例

図 11 のプログラムから得られた情報を表 6 に, その情報から作成された楽曲を図 12 に示す.

```
$fin = 10;
for ( $i = 0 ; $i <= $fin ; $i++ ) {
    for ( $j = $i ; $j >= 0 ; $j-- ) {
        if ( $j % 2 == 0 ) {
            print "#";
        }
        else {
            print "*";
        }
    }
    print "\n";
}
```

図 **11** 整形済み楽曲作成対象 Perl プログラム 1

表 **6** 図 11 の Perl プログラムの各行の情報

行	字下げ	コード	パーツ数	制御構造
1	0	C	4	other
2	0	C	14	for
3	1	G	14	for
4	2	Am	9	if
5	3	Dm	3	if
6	2	Am	1	if
7	2	Am	2	else
8	3	Dm	3	else
9	2	Am	1	else
10	1	G	1	for
11	1	G	3	for
12	0	C	1	for

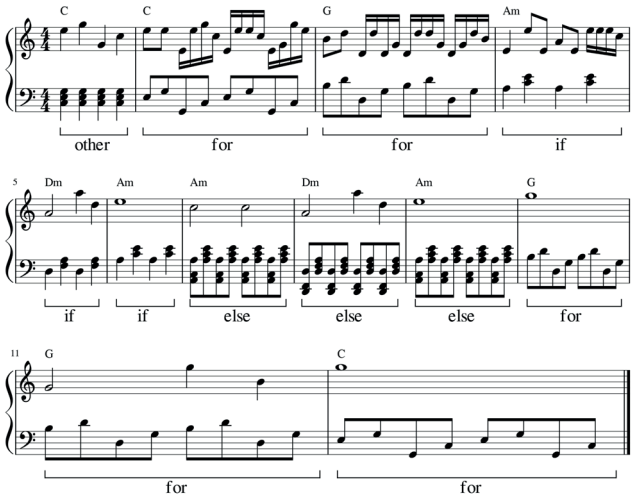

図 **12** 図 11 の Perl プログラムから作成された楽曲

図 13 のプログラムから得られた情報を表 7 に, その情報から作成された楽曲を図 14 に示す.

```
my $n = <STDIN>;
chomp $n;
my $i = 1;

while ( $i <= $n ) {
    if ( ( $i % 2 ) == 0 ) {
        $num[$i] = '2 の倍数';
    }
    elsif ( ( $i % 3 ) == 0 ) {
        $num[$i] = '3 の倍数';
    }
    else {
        $num[$i] = ' その他';
    }
    $i++;
}
&show(@num);

sub show {
    my @num = @_;
    my $n   = @num;

    for ( my $i = 1 ; $i < $n ; $i++ ) {
        print "$i:$num[$i]\n";
    }
}
```

図 **13** 整形済み楽曲作成対象 Perl プログラム 2

表 **7** 図 13 の Perl プログラムの各行の情報

行	字下げ	コード	パーツ数	制御構造
1	0	C	5	other
2	0	C	3	other
3	0	C	5	other
4	0	C	7	while
5	1	G	11	if
6	2	Am	4	if
7	1	G	1	if
8	1	G	11	elsif
9	2	Am	4	elsif
10	1	G	1	elsif
11	1	G	2	else
12	2	Am	4	else
13	1	G	1	else
14	1	G	3	while
15	0	C	1	while
16	0	C	2	other
17	0	$D\#$	3	other
18	1	$A\#$	5	other
19	1	$A\#$	5	other
20	1	$A\#$	15	for
21	2	Cm	3	for
22	1	$A\#$	1	for
23	0	$D\#$	1	other

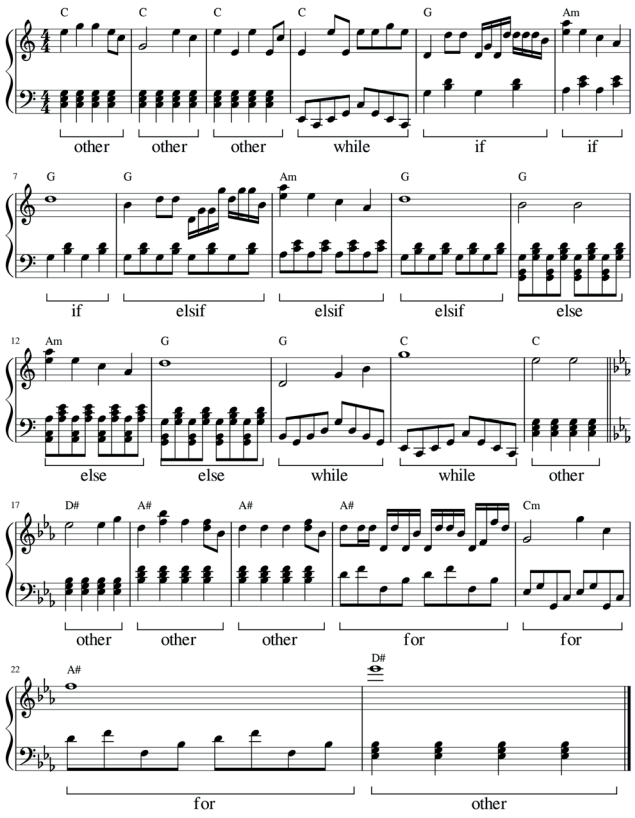

図 **14** 図 13 の Perl プログラムから作成された楽曲

図 11 のプログラムはネストの変化が多い．これより, コードの変化が多く, 楽曲が複雑になっていることがわかる．これに対して図 13 のプログラムはネストの変化が少ない．これより, コードの変化は少なく, 楽曲が単調になっていることがわかる.

また, どちらのプログラムも if, elsif, else の中の処理は似ているため, メロディは同じような変化をし, 伴奏だけが大きく変化していることがわかる.

図 13 のプログラムにはサブルーチンが存在する．これにより, サブルーチンが現れたときは 3 度上の嬰ニ長調に転調していることがわかる. これは, デフォルトの設定が 3 度上のスケールに転調するようになっているためである.

7. まとめ

Perl プログラムから楽曲を作成した．コード進行の検討の下, メロディや伴奏を決定し, 自然な楽曲作成を目指した.

参考文献

[1] 佐藤 和哉, 丸山 一貫, 寺田 実 : “CodeMusician:プログラム実行可聴化の試み”, 第 3 回エンターテイメントと認知科学シンポジウム (2009).
[2] 佐藤 和哉, 平井 重行, 丸山 一貴, 寺田 実 : “CodeDrummer: リズムに着目したプログラム可聴化システム,” インタラクション 2011, 2SCL-14, 情報処理学会 (2011).
[3] 大澤 範高 : “継承関係の可聴化,” 日本ソフトウェア科学会第 8 回インタラクティブシステムとソフトウェアに関するワークショップ, 近代科学社, (2000).
[4] D. B. Boardman : *et al.*, “LISTEN: A Tool to Investigate the Use of Sound for the Analysis of Program Behavior,” COMPSAC’95, pp.184–189 (1995).
[5] P. Vickers and J. Alty : “Siren songs and swan songs: Debugging with music,” *Communications of the ACM*, Vol.46, No.7, pp.86–93 (2003).
[6] Andreas Stefik, *et al.* : “Layered Program Auralization: Using Music to Increase Runtime Program Comprehension and Debugging Effectiveness,” *ICPC’06*, pp.89–93, (2006).
[7] 杉原 大和, 六沢 一昭 : “プログラムからの楽曲生成の試み”, 第 5 回エンターテイメントと認知科学シンポジウム (2011).
[8] 松本 若菜, 六沢 一昭 : “Perl プログラムからの音楽作成,”, 千葉工業大学情報工学科卒業論文 (2013).
[9] 島岡 譲 他 10 名 : “和声 理論と実習”, 株式会社音楽之友社 (1964).
[10] テキスト音楽「サクラ」, 入手先 ⟨http://oto.chu.jp/⟩.