A Model of Musical Motifs

Torsten Anders
Interdisciplinary Centre for Computer Music Research
University of Plymouth

Abstract

This paper presents a model of musical motifs for composition. It defines the relation between a motif's music representation, its distinctive features, and how these features may be varied. Motifs can also depend on non-motivic musical conditions (e.g., harmonic, melodic, or rhythmic rules). The model was implemented as a constraint satisfaction problem.

1 Introduction

Compositional aspects such as harmony and counterpoint have often been formalised and implemented successfully. For example, Pachet and Roy [2001] provide a survey of constrained-based harmonisation systems. A key aspect of such systems is the introduction of formal models of established musical concepts such as note pitches, pitch classes, scale degrees, chord roots and so forth.

At the end of their survey, Pachet and Roy [2001] point out: "However, what remains unsolved is the problem of producing musically nice or interesting melodies." I believe, in order to formalise melody composition we need to model important melodic concepts such motifs and their relations. A crucial aspect of the motif concept is the diversity of possible motifs and their variations. The motif definition of the New Grove clearly points out this diversity.

> A short musical idea, melodic, harmonic, rhythmic, or any combination of these three. A motif may be of any size, and is most commonly regarded as the shortest subdivision of a theme or phrase that still maintains its identity as an idea. [Drabkin]

Motifs have been modelled for music analysis. For example, Buteau and Mazzola [2000] model the similarity of motifs, including motifs of different lengths. However, a motif model for composition is missing (to my knowledge). Löthe [1999] proposes a system creating minuet melodies over a given harmonic progression. The author discusses the importance of motif variations, but does not present a formalisation. The constraint-based composition system OMRC [Sandred, 2003] and its successor PWMC[1] support the composition of pieces from pre-composed motifs. These systems allow the user to apply further constraints on the music (e.g., rhythmic and harmonic rules). However, motif transformations are severely restricted (e.g., only transpositions are permitted).

This research presents a model of musical motifs for composition. The model expresses the relation between a motif's music representation, its identity (often notated a vs. b, cf. [Schoenberg, 1967]), and how it is varied (a^1 vs. a^2). Various musical aspects (e.g., the rhythm, melody, or harmony) can define the identity of a motif. A motif can be transformed in many ways, while retaining its identity. The user may define that some transformations are regarded as variations, while others are not (compare changing the melodic contour with a mere transposition).

The model is implemented as part of the constraint-based composition system *Strasheela* [Anders, 2007].[2] Users define a set of motifs (by features characterising their identity), and a set of variations on these motifs. Rules on motific identity and variation can be applied. For example, a rule may constrain that a certain phrase consists of variations of the same motif, where the motif's identity is unknown in the definition. Additionally, users can constrain other aspects of the music. For example, harmonic, rhythmic, and formal rules are defined independently of the motif definition, but directly affect the motifs in the solution. For efficiency, Strasheela uses state-of-the-art constraint programming techniques: a constraint model based on the notion of computational spaces [Schulte, 2002] makes search strategies programmable.

Paper Outline

The rest of the paper is organised as follows. The motif model formalism is explained in Sec. 2. Section 3 demonstrates the model with two motifs from Beethoven's 5th symphony. The text concludes with a discussion (Sec. 4).

[1] Personal communication, PRISMA meeting, January 2007 in Montbéliard, France.

[2] Strasheela is available for download at `http://strasheela.sourceforge.net/`.

2 The Formal Model

The proposed motif model is stated as a constraint satisfaction problem (CSP). A CSP closely resembles a mathematical specification. A CSP imposes *constraints* (relations) between *variables* (unknowns), where each variable has a *domain* (a set of possible values). However, a CSP is also executable: modern constraint solvers efficiently find *solutions* for a CSP (i.e., determine each variable to a value of its domain which is consistent with all its constraints).

In this model, a motif is a tuple of the three variables *representation*, *description*, and *variation* (Fig. 1). The variable *representation* basically stores the information recorded in a music notation of the motif. For example, *representation* expresses the temporal organisation of notes in the motif and their pitches. Its domain is the set of all motif candidates. Please note that in an efficient implementation of the model, the representation is not a variable itself but it *contains* variables (e.g., all note pitches and durations in the *representation* may be variables). The model abstracts away from the actual music representation format: this information can be encoded in any hierarchic representation format which supports variables and an interface for accessing score information (e.g., a variant of CHARM [Harris et al., 1991], or Smoke [Pope, 1992] supporting variables). The model was implemented using the Strasheela music representation [Anders, 2007].

$$
\begin{aligned}
&\mathit{motif} ::= \langle \mathit{representation}, \mathit{description}, \mathit{variation} \rangle \\
&\mathit{representation} ::= \text{some hierarchic music representation} \\
&\mathit{description} ::= \langle \mathit{feature}_1 \colon \mathit{variable\ list}_1, \\
&\qquad\qquad \mathit{feature}_2 \colon \mathit{variable\ list}_2, \\
&\qquad\qquad \ldots \rangle \\
&\mathit{variation} ::= \mathit{motif} \mapsto (0 \vee 1) \\
&\mathit{makeVariation} ::= \{\mathit{feature}_1 \colon f_1 : \mathit{motif} \mapsto \mathit{variable\ list}_1, \\
&\qquad\qquad \mathit{feature}_2 \colon f_2 : \mathit{motif} \mapsto \mathit{variable\ list}_2, \\
&\qquad\qquad \ldots\} \mapsto \mathit{variation}
\end{aligned}
$$

Figure 1: A motif consists of its music representation, a symbolic description, and a variation function

The variable *description* symbolically states distinctive motif features.

Each domain value of this variable describes a motif with its own identity (e.g., one domain value describes motif a and another motif b). Because we have no agreed feature set which distinguishes the identity of a motif (cf. the Grove motif definition above), *description* can contain any information (e.g., the motif's note durations and its melodic intervals). *description* can have an arbitrary format, but a consistent format of its domain values simplifies the CSP definition. The following format combines flexibility with convenience: *description* is a tuple of feature-value pairs (Fig. 1). A feature is a descriptive label (e.g., *durations*) and its value is a list of (often determined) variables (e.g., the note durations for motif a).

The variable *variation* denotes a specific motif variation. The *variation* domain consists of functions which map a motif to a Boolean variable (Fig. 1). Please note that this texts notates the domain of a variable simply as a disjunction ($\vee$). A *variation* function imposes arbitrary constraints between the motif's *representation* and its *description* – if and only if the function returns 1 (i.e., *true*). The model enforces that only the selected *variation* returns 1 for a given motif instance. This approach is highly generic, but the *variation* functions can be complex to define. In a still flexible but more convenient approach, *variation* functions are created by the function *makeVariation*. *makeVariation* expects a tuple of feature-value pairs, where the features correspond to the features of the *description*, and their values are functions mapping a motif instance to a list of variables (e.g., a function returning the note durations of a motif). Please note that *makeVariation* unifies this list with the corresponding list in the selected motif *description*. For example, a model instance may constrain the note durations in the motif's *representation* to be equal to the durations in the *description*. This affects which domain values are selected for these variables.[3]

Figure 2 summarises the relations between all variables of the model. *myMotif* is any motif instance in the *score* (a subsection or a whole piece). The model's essence is highlighted in bold font.[4] For brevity, the definition of *makeVariation* is omitted.

[3]In the implementation, *description* and *variation* are encoded by finite domain integers. They point as indices in the respective domains. Selection constraints [Duchier et al., 1998] care for efficient constraint propagation.

[4]The function *map* applies the given function f to every element of the *variation*'s domain and returns the collected results.

$$
\begin{aligned}
&\forall\, \mathit{myMotif} \in \mathit{score} : \\
&\quad \exists\, \mathit{representation}, \mathit{description}, \mathit{variation} : \\
&\qquad \mathit{representation} = \bigvee \mathit{representation}_1, \ldots, \mathit{representation}_n \\
&\quad \wedge\ \mathit{description} = \bigvee \mathit{description}_1, \ldots, \mathit{description}_n \\
&\quad \wedge\ \mathit{variation} = \bigvee \mathit{variation}_1, \ldots, \mathit{variation}_n \\
&\quad \wedge\ \boldsymbol{myMotif} = \langle \boldsymbol{representation}, \boldsymbol{description}, \boldsymbol{variation} \rangle \\
&\quad \wedge\ \mathbf{1} = \sum \boldsymbol{map}(\boldsymbol{getInitialDomain}(\boldsymbol{variation}), \\
&\qquad\qquad\qquad \boldsymbol{f} : \boldsymbol{f}(\boldsymbol{v}) := \boldsymbol{v}(\boldsymbol{myMotif})) \\
&\quad \wedge\ \boldsymbol{variation}(\boldsymbol{myMotif}) = \mathbf{1}
\end{aligned}
$$

Figure 2: Relations between the motif model variables (essence in bold font)

3 An Example

This section models well-known motifs from the first movement of Beethoven's Fifth Symphony as an example. Figure 3 classifies some motif instances according to motif identity and variation. The presented classification allows for considerable mutability of the first variation of motif *a*, but other classifications can be expressed with this model as well. A set of motifs and their classification is modelled by defining domains for the three variables *representation*, *description*, and *variation*. The set of solutions for a single motif instance includes all motifs shown in Fig. 3 – among similar motifs. However, additional rules can further restrict the music (e.g., rhythmic, harmonic, and contrapuntal rules), and many motif instances can be part of a CSP.

The *representation* domain consists of note sequences, where each note in a sequence has parameter values for its *duration* and *pitch* (Fig. 4).[5] As these parameters can have any value, all shown Beethoven motifs are members of this domain. Please note that instances of motif *a* and *b* differ in length: the motif length is not fixed in *representation*.[6]

The *description* domain characterises rhythmic and melodic features

[5]The pause is not modelled for simplicity. It can be addressed by a note offset parameter [Anders, 2007].

[6]The implementation encodes all motif instances with the same – maximum – length internally. Notes are marked as 'non-existing' by setting their duration to 0 [Anders, 2007].

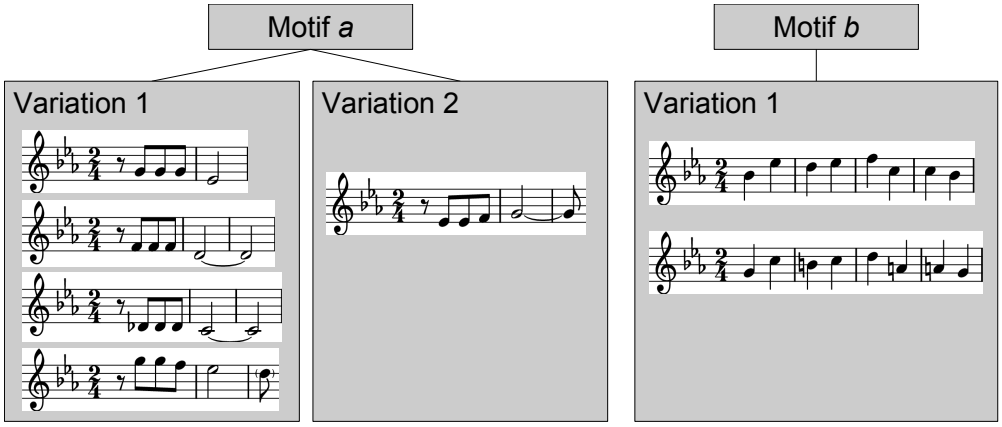

Figure 3: Motifs from Beethoven's Symphony No. 5

which distinguish the two Beethoven motifs a and b. Please note that the feature sets differ between motifs: $description_a$ specifies the *pitchContour* (the sequence of pitch interval directions), whereas $description_b$ specifies *scaleDegreeIntervals* (the sequence of distances between note pitches measured in scale degrees). Also, note that $description_a$ makes use of variables (e.g., the last note duration is not fixed).

Finally, the functions in the *variation* domain constrain the relation between the *representation* and the *description* of a motif instance. The functions *getNoteDurations*, *getPitchContour*, and *getScaleDegreeIntervals* access the motif's *representation*. For example, *getNoteDurations* can be implemented as shown in (1), where *getNotes* returns the notes in the motif's *representation*, and *getDuration* returns the duration of a note. Please remember that *makeVariation* unifies the variable list returned by the these functions with the corresponding variable list in the description. *description* values can differ in their set of features (see above): variations only constrain those motif aspects specified by the *description* of a motif (e.g., $variation_1$ does not constrain the pitch contour in case the motif's description is $motif_b$). $variation_2$ inverses the pitch contour of a motif (cf. Fig. 3), but $variation_2$ is only permitted for motif a.

$$getNoteDurations(myMotif) := map(getNotes(myMotif), getDuration) \tag{1}$$

$$
\begin{aligned}
\mathit{representation} := \bigvee\ & \mathit{sequence}_1 \text{ with notes of specific duration and pitch,} \\
& \mathit{sequence}_2 \text{ with notes of specific duration and pitch,} \\
& \ldots \\
\mathit{description}_a := \langle & \mathit{durations}\colon (\eighthnote, \eighthnote, \eighthnote, (\bigvee \halfnote, \ldots, \wholenote)) \\
& \mathit{pitchContour}\colon (\rightarrow, (\bigvee \rightarrow, \searrow), \searrow)\rangle \\
\mathit{description}_b := \langle & \mathit{durations}\colon (\dottedquarternote, \dottedquarternote, \dottedquarternote, \dottedquarternote, \dottedquarternote, \dottedquarternote, \dottedquarternote, \dottedquarternote), \\
& \mathit{scaleDegreeIntervals}\colon (3, -1, 1, 1, -3, 0, -1)\rangle \\
\mathit{description} := \bigvee\ & \mathit{description}_a, \mathit{description}_b, \ldots \\
\mathit{variation}_1 := \langle & \mathit{durations}\colon \mathit{getNoteDurations}, \\
& \mathit{pitchContour}\colon \mathit{getPitchContour}, \\
& \mathit{scaleDegreeIntervals}\colon \mathit{getScaleDegreeIntervals}\rangle \\
\mathit{variation}_2 := \langle & \mathit{durations}\colon \mathit{getNoteDurations}, \\
& \mathit{pitchContour}\colon\ f : f(\mathit{myMotif}) := \\
& \qquad\qquad \mathit{getDescription}(\mathit{myMotif}) = \mathit{description}_a \\
& \qquad\qquad \wedge\ \mathit{inverse}(\mathit{getPitchContour}(\mathit{myMotif}))\rangle \\
\mathit{variation} := \bigvee\ & \mathit{makeVariation}(\mathit{variation}_1), \mathit{makeVariation}(\mathit{variation}_2), \ldots
\end{aligned}
$$

Figure 4: Definition of the three variables *representation*, *description*, and *variation* which model the Beethoven motifs

4 Discussion

This paper presented a motif model as a CSP which specifies the relation between the motif's music representation, a description of distinctive motif features, and motif variation definitions. The model was designed for computer-aided composition, but it can also be used as an executable representation of a motivic analysis. This research does not propose a new concept of motivic similarity, but allows for the application of various similarity models (e.g., the pitch contour). The model does not express a degree or genealogy of variations. However, it supports various additional cases. Non-motivic sections can be modelled by a *variation* function which does

not apply any constraint at all.[7] Contrapuntal motif combinations (e.g., a fugue subject) can be search for by constraining multiple motif instances to the same *description*, but leaving feature values in the *description* itself undetermined in the definition. Overlapping motifs are possible if the music representation supports such nesting. Finally, higher-level formal relations can be expressed by nesting 'motif' instances (e.g., a theme may contain a motif sequence, and is specified by the theme's *description* and constrained by its *variation*).

References

Anders, T. (2007). *Composing Music by Composing Rules: Design and Usage of a Generic Music Constraint System*. Ph. D. thesis, School of Music & Sonic Arts, Queen's University Belfast.

Buteau, C. and G. Mazzola (2000). From Contour Similarity to Motivic Topologies. *Musicae Scientiae 4*(2).

Drabkin, W. Motif. In L. Macy (Ed.), *Grove Music Online ed.* (Accessed 9 August 2007), http://www.grovemusic.com.

Duchier, D., C. Gardent, and J. Niehren (1998). *Concurrent Constraint Programming in Oz for Natural Language Processing*. Universität des Saarlandes, Germany: Programming Systems Lab.

Harris, M., A. Smaill, and G. Wiggins (1991). Representing Music Symbolically. In *IX Colloquio di Informatica Musicale*, Genoa, Italy.

Löthe, M. (1999). Knowledge Based Automatic Composition and Variation of Melodies for Minuets in Early Classical Style. In W. Burgard, T. Christaller, and A. B. Cremers (Eds.), *KI-99: Advances in Artifical Intelligence: 23rd Annual German Conference on Artificial Intelligence*. Springer.

Pachet, F. and P. Roy (2001). Musical Harmonization with Constraints: A Survey. *Constraints Journal 6*(1).

Pope, S. T. (1992). The Smoke Music Representation, Description Language, and Interchange Format. In *Proceedings of the International Computer Music Conference*, San Jose.

[7] To eliminate symmetries (i.e., different solutions which are equivalent), this non-motivic variation should determine the motif description to some domain value.

Sandred, O. (2003). Searching for a Rhythmical Language. In *PRISMA 01*. Milano: EuresisEdizioni.

Schoenberg, A. (1967). *Fundamentals of Musical Composition*. London: Faber and Faber.

Schulte, C. (2002). *Programming Constraint Services*. Springer.