Introducing a Context-based Model and Language for Representation, Transformation, Visualization, Analysis and Generation of Music

David M. Hofmann
University of Music Karlsruhe
hofmann@hfm.eu

ABSTRACT

A software system for symbolic music processing is introduced in this paper. It is based on a domain model representing compositions by means of individual musical contexts changing over time. A corresponding computer language is presented which allows the specification and textual persistence of composition models. The system provides an infrastructure to import, transform, visualize and analyze music in respect of individual musical aspects and parameters. The combination of these components provides the basis for an automated composition system capable of generating music according to given statistical target distributions using an evolutionary algorithm.

1. INTRODUCTION

The goal of the presented research project is to yield new findings related to musical composition processes by developing a software system capable of processing and generating music. In order to perform this complex task in a sophisticated manner, further components are necessary: models for music representation, facilities to analyze compositions and an infrastructure to transform results into accurate presentation formats for the user. These modules have proved efficient for symbolic music processing, computer-aided musicology and visualization purposes. This paper demonstrates several applications of the individual components and how they can be combined to tackle the automated composition challenge.

2. MOTIVATION

While music is traditionally notated, read and analyzed in scores, the proposed system provides alternative models to represent music in which individual musical aspects are encoded separately. In the domain of symbolic music processing, musical compositions are often represented as a sequence of notes and rests. While this might be sufficient for a number of use cases, a more complex model is required for extensive musical analysis and for understanding relations between individual aspects. While a single note or sound itself does not have a great significance, its meaning becomes apparent when considering various musical contexts. These include: metric context, rhythm, key, tonal center, harmonic context, harmonic rhythm, scale, pitch, loudness and instrumentation. It is proposed that musical compositions can be represented as a set of the named parameters changing over time.

Copyright: ©2016 David M. Hofmann et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License 3.0 Unported, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

3. COMPOSITION MODEL

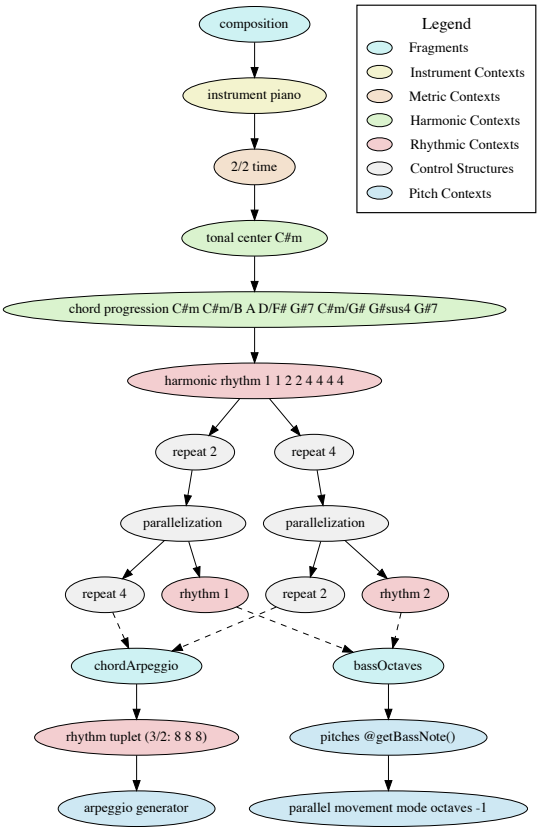

Figure 1. Manually specified context model of the first four measures of Beethoven's *Piano Sonata No. 14, Op. 27 No. 2*, first movement

Models are based on a tree structure containing musical contexts and may further contain so called *modifiers*, *generators* and *control structures*. Modifiers specify how already existing musical contexts are altered in the course of the composition (e.g. transpositions or rhythmic adjustments). Generators are used to create new contexts based on already existing ones (e.g. arpeggios based on chords). Consider Figure 1, which shows a context model of the first four measures of Beethoven's *Piano Sonata No. 14, Op. 27 No. 2*, commonly known as *Moonlight Sonata*.

Each context model has a root node labeled *composition*. Below this node all model elements may be arranged arbitrarily, i.e. it is not prescribed which elements appear on which hierarchy level in the model. In the example, an instrumentation context (*piano*), a metric context (*2/2 time*) and the key (*C#m*) are specified. The model also supports the specification of chord progressions. To supply the durations of each chord, a harmonic rhythm context is required.

Control structures such as repetitions may be nested recursively in order to reflect repeating structures at any level of the composition. For example, in the given model the first repetition branch (*repeat 2*) represents the first two measures. In each measure, the right hand plays four eighth triplets (nested *repeat 4*). Parallel voices are modeled using a control structure named *parallelization*. Otherwise multiple nodes on the same level are interpreted as context sequences. If a subtree contains the same element type on different hierarchy levels, the lower-level context overwrites the higher-level context, allowing to overwrite contexts temporarily.

Models may be split up into so called fragments, which are subtrees that may be referenced from elsewhere. In this way compositions can be specified in a redundancy-free manner since any context combination needs to be specified only once. In the presented example, a fragment named *chordArpeggio* is referenced twice as well as a fragment called *bassOctaves*. The former demonstrates the usage of an *arpeggioGenerator*. It computes a specific chord inversion based on the context harmony and cycles through its pitches in a given sequence. The fragment *bassOctaves* contains a pitch context and a modifier. The pitch context is fed by an expression (prefixed with @) invoking a function called *getBassNote()*, which provides the current bass note of the context harmony. The *parallelMovement* modifier turns single pitches into simultaneously played octaves.

The model captures higher-level concepts which are relevant for compositions such as hierarchical relations and nestings, harmonic progressions on multiple levels and descriptions on how musical material is derived and developed. Instead of enumerating notes, the model is also capable of describing the way music is derived from higher-level building blocks. This aligns with the composition process of human individuals, who generally think in higher-level concepts and structures, many of which this model tries to accommodate. However, a complete list and description of all context types, modifiers, generators and control structures is not possible due to space limitations. Up to now the model was used for representing and processing western tonal music and percussion music, yet it was designed bearing in mind that its application could be extended to atonal music, music from non-western cultures or even electroacoustic music.

Note that there are multiple possible context models for one and the same composition. They can be of explicit nature (comparable to a hierarchical, redundancy-optimized score) or of implicit nature (describing musical development processes). A module for the automatic construction of context models for existing compositions in MIDI or MusicXML format is currently under development. Context models can also be specified and edited manually as explained in the following section.

4. DOMAIN-SPECIFIC COMPOSITION LANGUAGE

```
composition
{
  instrument piano
  {
    time 2/2
    {
      tonalCenter C#m
      {
        chordProgression C#m C#m/B A D/F# G#7 C#m/G# G#sus4 G#7
        {
          harmonicRhythm 1 1 2 2 4 4 4 4
          {
            repeat 2
            {
              parallel
              {
                repeat 4
                {
                  fragmentRef chordArpeggio
                }
                rhythm 1
                {
                  fragmentRef bassOctaves
                }
              }
            }
            repeat 4
            {
              parallel
              {
                repeat 2
                {
                  fragmentRef chordArpeggio
                }
                rhythm 2
                {
                  fragmentRef bassOctaves
                }
              }
            }
          }
        }
      }
    }
  }
}
fragment chordArpeggio
{
  rhythm (3/2 : 8 8 8)
  {
    arpeggioGenerator startInversion 2 startOctave 3
      noteIndexSequence 0 1 2
  }
}
fragment bassOctaves
{
  pitches relative to harmony startOctave 3 findNearestOctave
      true @getBassNote()
  {
    parallelMovement mode octaves -1
  }
}
```

Listing 1. Syntactical representation of the composition model shown in Figure 1

A domain-specific composition language corresponding to the introduced context model allows the textual specification of compositions. This process can also be reversed: algorithmically generated models can be persisted in human-readable text files. Related structured music description languages are *SARAH* [1] and the Hierarchical Music Specification Language [2]. An example is given in Listing 1, which is equivalent to the model in Figure 1. The language infrastructure was built using the framework Xtext [3]. Based on the provided components, an Eclipse-based Integrated Development Environment (IDE) was developed for the composition language featuring syntax highlighting, hyperlinking, folding, outline view and automatic code completion. User interfaces for other features described in the following sections are also integrated.

<table>
<tr><td rowspan="7">Stream 1</td><td>Instrument (1)</td><td colspan="48">piano</td></tr>
<tr><td>Meter (4)</td><td colspan="12">2/2 time</td><td colspan="12">2/2 time</td><td colspan="12">2/2 time</td><td colspan="12">2/2 time</td></tr>
<tr><td>Tonal Center (1)</td><td colspan="48">C#m</td></tr>
<tr><td>Harmony (8)</td><td colspan="12">C#m</td><td colspan="12">C#m/B</td><td colspan="6">A</td><td colspan="6">D/F#</td><td colspan="3">G#7</td><td colspan="3">C#m/G#</td><td colspan="3">G#sus4</td><td colspan="3">G#7</td></tr>
<tr><td>Harmonic Rhythm (8)</td><td colspan="12">1</td><td colspan="12">1</td><td colspan="6">2</td><td colspan="6">2</td><td colspan="3">4</td><td colspan="3">4</td><td colspan="3">4</td><td colspan="3">4</td></tr>
<tr><td>Rhythm (48)</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td><td>12</td></tr>
<tr><td>Pitches (48)</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>E</td><td>A3</td><td>C#</td><td>E</td><td>A3</td><td>C#</td><td>E</td><td>A3</td><td>D</td><td>F#</td><td>A3</td><td>D</td><td>F#</td><td>G#3</td><td>B#3</td><td>F#</td><td>G#3</td><td>C#</td><td>E</td><td>G#3</td><td>C#</td><td>D#</td><td>F#3</td><td>B#3</td><td>D#</td></tr>
<tr><td rowspan="7">Stream 2</td><td>Instrument (1)</td><td colspan="48">piano</td></tr>
<tr><td>Meter (4)</td><td colspan="12">2/2 time</td><td colspan="12">2/2 time</td><td colspan="12">2/2 time</td><td colspan="12">2/2 time</td></tr>
<tr><td>Tonal Center (1)</td><td colspan="48">C#m</td></tr>
<tr><td>Harmony (8)</td><td colspan="12">C#m</td><td colspan="12">C#m/B</td><td colspan="6">A</td><td colspan="6">D/F#</td><td colspan="3">G#7</td><td colspan="3">C#m/G#</td><td colspan="3">G#sus4</td><td colspan="3">G#7</td></tr>
<tr><td>Harmonic Rhythm (8)</td><td colspan="12">1</td><td colspan="12">1</td><td colspan="6">2</td><td colspan="6">2</td><td colspan="3">4</td><td colspan="3">4</td><td colspan="3">4</td><td colspan="3">4</td></tr>
<tr><td>Rhythm (6)</td><td colspan="12">1</td><td colspan="12">1</td><td colspan="6">2</td><td colspan="6">2</td><td colspan="6">2</td><td colspan="6">2</td></tr>
<tr><td>Pitches (6)</td><td colspan="12">[C#3 C#2]</td><td colspan="12">[B2 B1]</td><td colspan="6">[A2 A1]</td><td colspan="6">[F#2 F#1]</td><td colspan="6">[G#2 G#1]</td><td colspan="6">[G#2 G#1]</td></tr>
<tr><td></td><td>Time (Measures)</td><td colspan="12">1</td><td colspan="12">2</td><td colspan="12">3</td><td colspan="12">4</td></tr>
</table>

Figure 2. Stream model of the first four measures of Beethoven's *Piano Sonata Op. 27 No. 2*, first movement.

5. STREAM MODEL

An alternative representation of musical compositions is provided by so-called *stream models*. These contain parallel timelines for individual musical aspects revealing each musical dimension separately. Stream models provide a novel visual presentation of music suitable for scientific analysis, musicological examination and educational purposes. In Figure 2, a stream model equivalent to the context model in Figure 1 is presented. A context model is transformed into a stream model by means of a compiler. It walks through the tree while resolving references, applying modifiers and generators and finally aggregating contexts at the leaf nodes resulting in a set of separate streams for each musical aspect. Stream models can be considered as expanded context models.

6. SCORE TRANSFORMATION

The infrastructure also provides the functionality to convert stream models into human-readable scores. This is achieved by another compiler, which currently supports LilyPond [4] output resulting in corresponding MIDI and PDF files. An example score is shown in Figure 9. The LilyPond export is configurable, allowing users to render traditional scores and even lead sheets containing chord symbols and fret board diagrams. Another export module for MusicXML [5] output is planned.

7. ANALYSIS FRAMEWORK

An integral part of the system is an analysis framework suitable for extracting statistical data from musical pieces. A related toolkit for computer-aided musicology named *music21* was developed by Cuthbert and Ariza [6]. The proposed system is based on an extensible, module-based architecture. Each module implements specific analysis algorithms for specific musical aspects. The extracted data is written to comma-separated value (CSV) files. Note that different scopes of analyses are possible: corpus, single piece, individual voice in a piece and section-wise analysis. The modules are described in the following sections.

7.1 Duration and Rhythmic Context Analysis

This module analyzes note and rest duration distributions and the duration ratio of notes and rests. Durations can also be analyzed with respect to other musical parameters. For example, Figure 3 illustrates the analysis of note durations

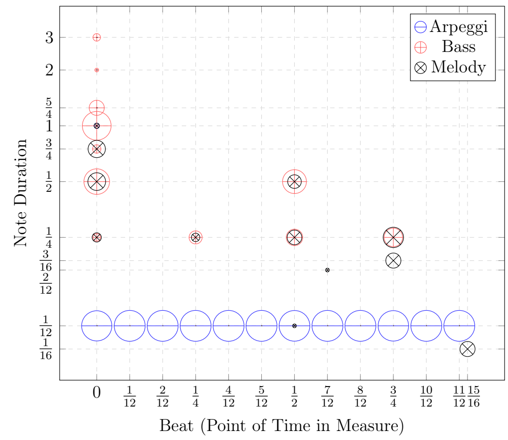

Figure 3. Note duration distribution analysis depending on beats of the complete first movement of Beethoven's *Piano Sonata Op. 27 No. 2*. Circle areas are proportional to the quantity of the data pairs. Triads of the arpeggio notes are equally distributed throughout the measures. The bass and melody voice have selective accentuations, mostly on the first beat of a measure and the following quarter beats. The diagram also reveals that there are certain rhythmic similarities between the melody and the bass voice.

depending on the point of time in the measure they occur for individual voices. With the help of the diagram, rhythmic similarities between voices are visualized. Figure 4 shows an aggregated histogram of note onset times in corresponding measures for various composers based on the analysis of 1093 pieces in MusicXML format. The corpus was compiled by the author from various Internet sources, mainly musescore.org.

7.2 Interval Leap Analysis

This module analyzes the distribution of interval leaps between successive notes in a stream. An example histogram is shown in Figure 7. In case of different note counts (e.g. single notes next to chords or chords with different numbers of notes) the system selects a set of interval leaps with the minimum absolute distance.

7.3 Dissonance Analysis

This module analyzes a piece concerning consonance and dissonance. The dissonance of two simultaneously sounding notes is calculated by taking the frequency proportions of the fundamentals into account. For example, the interval of a fifth is characterized by the frequency ratio 3:2. The so

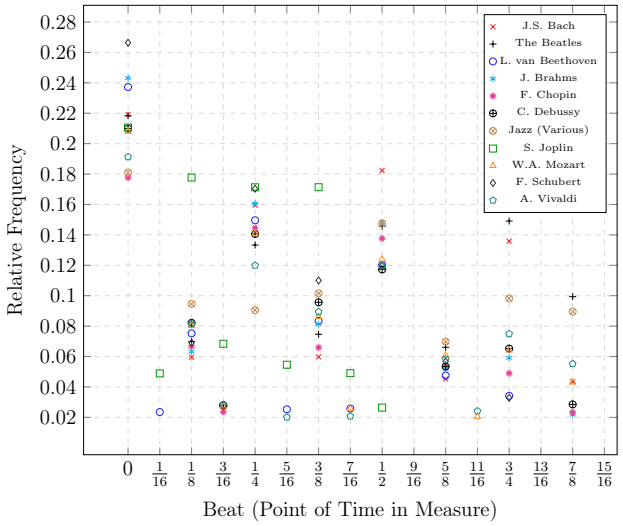

Figure 4. Aggregated beat distribution for various composers. Each analyzed composer places most of the notes on the first beat of a measure. The probability of syncopation increases with more modern music. Values lower than 0.02 were filtered out for better clarity.

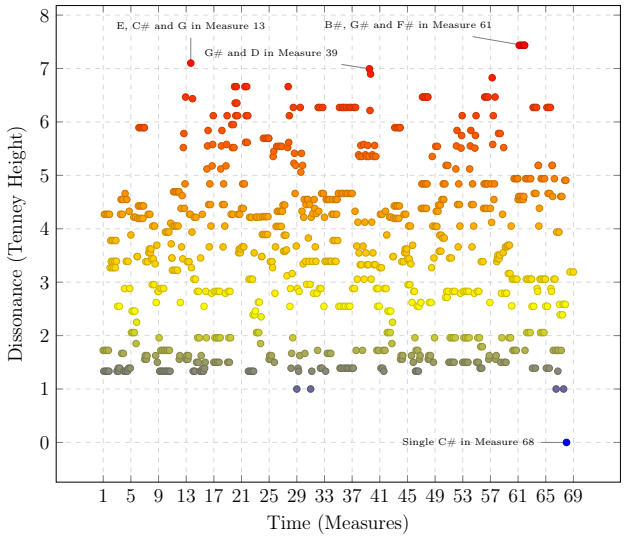

Figure 5. Dissonance analysis of the complete first movement of Beethoven's *Piano Sonata Op. 27 No. 2*

called Tenney Height, which represents a scalar dissonance value for two given tones, is computed using the formula $log_2 ab$, where a and b are nominator and the denominator of the ratio (in this example: $log_2 6 \approx 2.59$) [7]. The dissonance value of a chord consisting of more than two notes is obtained by computing the average dissonance of all note combinations. Figure 5 shows a dissonance plot of the first movement of the *Moonlight Sonata*.

7.4 Harmonic Analysis

The harmonic analysis module focuses on simultaneously sounding notes in order to determine the histograms of keys and harmonies used in the composition. The module also computes a *chord compliance ratio* by dividing the number of notes which are part of the context harmony by the total number of notes. This value is typically high for accompanying voices providing context chords. Moreover the *scale compliance* is analyzed, which is defined as the ratio between notes belonging to the context scale divided by the total number of notes. This is analyzed with respect to the context harmony and the tonal center. Tonally simple and coherent pieces are typically characterized by high scale compliance ratios. The module also supports the analysis and visualization of chord progressions in a directed graph as shown in Figure 6.

8. EVOLUTIONARY COMPOSITION GENERATION

The previous section was concerned with extracting statistical data from existing compositions. For music generation, this process is reversed: statistical distributions are given as input and the system generates compositions that adhere to these requirements as accurately as possible. The space of representable compositions is huge, so a brute-force search would not yield accurate results in an acceptable time. Therefore, so called evolutionary algorithms are used. Programs of this kind have successfully been applied to specific musical problems such as evolving jazz solos [8], rhythms [9], chord harmonization [10, 11] and automated composition systems [12, 13, 14].

8.1 Algorithm Specification

The system creates an initial generation of compositions randomly. Every model is evaluated by compiling it to a stream model (explained in section 5) and analyzing it statistically (see section 7). All distributions are compared with the desired input distributions and all absolute deviations are added up. The goal of the evolutionary process is to minimize the total deviation to zero, effectively implementing a multi-objective optimization [15]. This is achieved by recombining subtrees of compositions selected considering their fitness measure. Mutations are performed by adding, modifying or removing nodes or subtrees. This technique can be considered a special form of *Genetic Programming* [16]. The algorithm benefits from the design of the context model (described in Section 1), as "composing" is now no longer a matter of concatenating notes, but a matter of assembling and restructuring context trees, in which each individual aspect of the music is accessible separately. Results are persisted in text files (the syntax of which was introduced in section 4) and are subsequently transformed to scores (see section 6).

8.2 Algorithm Input and User-defined Constraints

Additional constraints to limit the search space may be supplied optionally in form of an initial context model. It may contain fragments to be incorporated into the composition or predefined constraints such as time signatures, instruments or chord progressions. For this purpose, nodes or subtrees can be marked as *fixed* indicating that they are not to be modified during the evolutionary process.

As an example, it is demonstrated how a blues composition can be generated. The model is shown in figure 8. The predefined nodes specify a basic twelve bar blues pattern as a constraint space. Two parallel voices are defined, one of which is an accompaniment and the other one is the lead voice. The accompaniment is defined in a separate

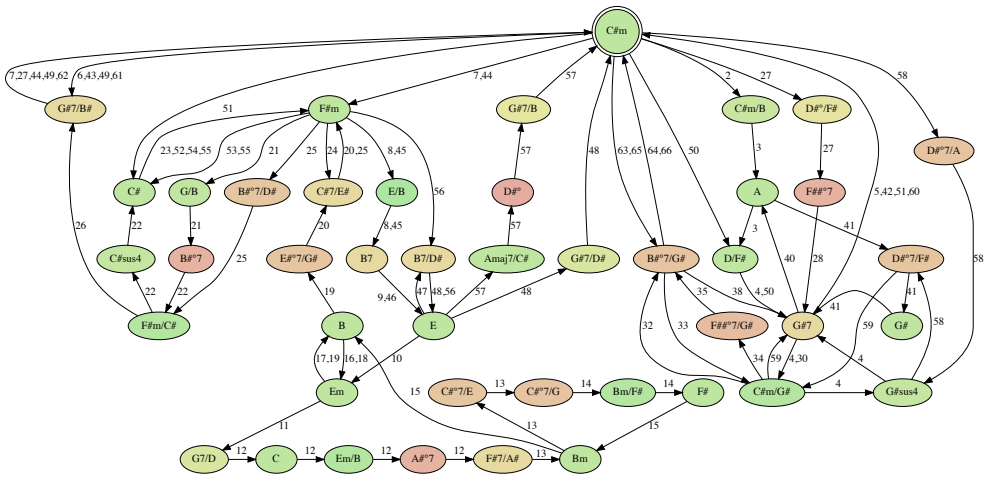

Figure 6. Chord progression graph of the complete first movement of Beethoven's *Piano Sonata Op. 27 No. 2*. The numbers identify the measures in which the respective chord transition was detected. The colors of the nodes correlate with the dissonance value of the chord (green: consonant; red: dissonant).

fragment and consists of alternating fifths and sixths. Literal numbers in pitch contexts are interpreted as degrees on the context scale. Lastly, a final chord (C^7) is predefined. The lead voice is generated by the evolutionary algorithm.

Besides the optional initial context model, the algorithm requires statistical target distributions as input. The following target distributions were set: The desired note duration ratio of the lead voice was set to 95%, meaning that only 5% of the generated material should consist of rests. The requested scale compliance ratio was set to 100%. Furthermore, two target distributions were supplied. The first distribution demands that about 30% of the generated notes should be quarter notes, 50% eighth notes and 20% sixteenth notes. Additionally, the interval leap distribution shown in Figure 7 was given as a target. Of course, more complex combinations of statistical target distributions may be used.

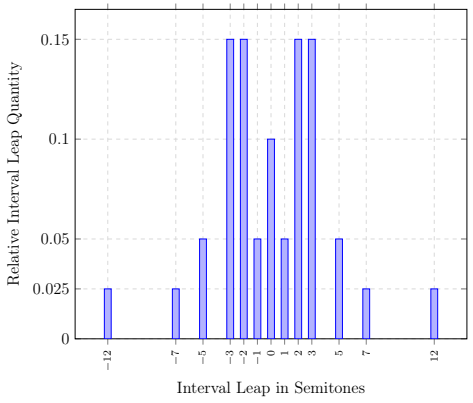

Figure 7. Symmetric Target Distribution for Interval Leaps

9. RESULTS

The computer automatically generated the model shown in Figure 8. The corresponding score is shown in Figure 9. Statistical target distributions and deviations are listed in Table 1. Most listeners found this short piece to be musically pleasant and entertaining. Overall, generated compositions are musically appealing at times. Nonetheless, this is not guaranteed even when using the same set of input parameters repeatedly. The rate of pleasant compositions depends on the number of target distributions and the number of initially predefined constraints. When specifying too few targets, the divergence of both the musical style and the musical quality increases.

Fitness Function	**Target**	**Distance**
Note Duration Ratio	0.95	0.005
Scale Compliance	1.0	0
Note Durations	30% quarter notes 50% eighth notes 20% sixteenth notes	0.1
Interval Leaps	see Figure 7	0.27
Total	0	0.375

Table 1. Fitness functions, target values and deviations from the optimum values for the generated blues composition

Currently the implementation is only capable of optimizing against target features computed for the whole piece resulting in rather monotonous music. The goal is to extend the system in such a way that individual sections of the piece are optimized against a set of section-wise defined target distributions. This could potentially result in a system generating interesting and diverse music.

Another limitation is that the algorithm often does not find an optimal solution, as the search space is very large. Another reason for this might be that mutation and crossover operators still need to be improved. Even though the generated solutions might not satisfy all statistical criteria in all cases, they nonetheless can be appealing and interesting. Eventually, the fact that statistical expectations are not entirely met can contribute to a certain naturalness of a computer-generated composition, causing unexpected musical twists and variety in the music.

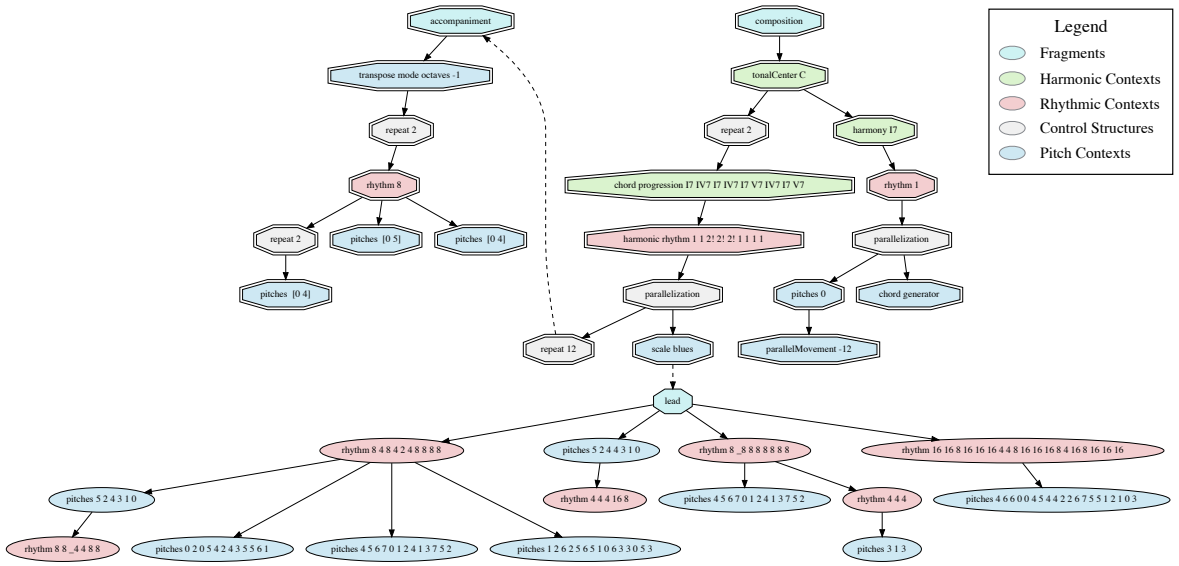

Figure 8. Context model resulting from an evolutionary composition process. Nodes surrounded by octagons were predefined. The subtree below the *lead* node was generated by the evolutionary algorithm.

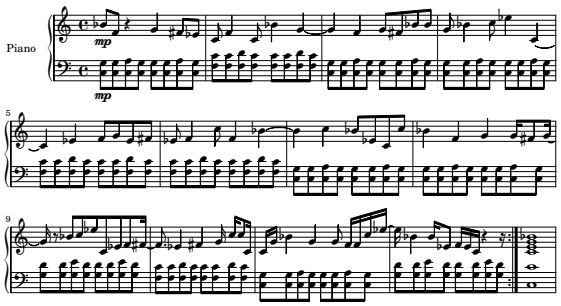

Figure 9. Score of the generated blues composition

10. CONCLUSIONS AND FUTURE WORK

A software system for symbolic music processing was introduced and the functionality of its components was explained for various use cases including context-based music representation, transformation and statistical analysis. Furthermore the combination of these components to form an automated composition system was demonstrated. Future research will be conducted regarding combinations of statistical distributions resulting in appealing musical outcomes. The system will be further improved by providing the functionality to supply different sets of target distributions for multiple sections of a piece. The goal is to develop a higher-level algorithm creating section-wise target distributions and then use the proposed algorithm to generate musical material. Another future goal is to develop a graphical user interface for the composition module allowing users to specify criteria of the desired musical output.

Acknowledgments

This research is generously funded by a scholarship granted by the state of Baden-Württemberg. The author would like to thank his supervisor Prof. Dr. Thomas Troge, the five anonymous reviewers and the paper chair Hans Timmermans for their support and valuable feedback on this paper.

11. REFERENCES

[1] C. Fox, "Genetic hierarchical music structures," in *Proceedings of the 19th International FLAIRS Conference*. AAAI Press, 2006, pp. 243–247.

[2] L. Polansky, P. Burk, and D. Rosenboom, "Hmsl (hierarchical music specification language): A theoretical overview," *Perspectives of New Music*, vol. 28, no. 2, pp. 136–178, 1990.

[3] S. Efftinge and M. Völter, "oaw xtext: A framework for textual dsls," in *Workshop on Modeling Symposium at Eclipse Summit*, vol. 32, 2006, p. 118.

[4] H.-W. Nienhuys and J. Nieuwenhuizen, "Lilypond, a system for automated music engraving," in *Proceedings of the XIV Colloquium on Musical Informatics (XIV CIM 2003)*, vol. 1, 2003.

[5] M. Good, "Musicxml for notation and analysis," *The virtual score: representation, retrieval, restoration*, vol. 12, pp. 113–124, 2001.

[6] M. S. Cuthbert and C. Ariza, "Music21: A toolkit for computer-aided musicology and symbolic music data," in *Proceedings of the 11th International Society for Music Information Retrieval Conference*, Utrecht, The Netherlands, August 9-13 2010, pp. 637–642.

[7] M. Deza and E. Deza, *Encyclopedia of Distances*. Springer, 2013.

[8] J. Biles, "Genjam: A genetic algorithm for generating jazz solos," in *Proceedings of the 1994 International Computer Music Conference*. San Francisco: ICMA, 1994, pp. 131–137.

[9] D. Horowitz, "Generating rhythms with genetic algorithms," in *Proceedings of the 1994 International Computer Music Conference*. San Francisco, CA: ICMA, 1994, pp. 142–143.

[10] R. McIntyre, "Bach in a box: The evolution of four part baroque harmony using the genetic algorithm," in *Proceedings of the IEEE Conference on Evolutionary Computation*, vol. 14(3). New York: IEEE Press, 1994, pp. 852–857.

[11] A. Horner and L. Ayers, "Harmonization of musical progressions with genetic algorithms," in *Proceedings of the 1995 International Computer Music Conference*, San Francisco, 1995, pp. 483–484.

[12] A. Horner and D. Goldberg, "Genetic algorithms and computer-assisted music composition," San Mateo, CA, pp. 437–441, 1991.

[13] B. Jacob, "Composing with genetic algorithms," in *Proceedings of the 1995 International Computer Music Conference*. San Francisco, CA: ICMA, 1995, pp. 452–455.

[14] ——, "Algorithmic composition as a model of creativity," *Organised Sound*, vol. 1(3), pp. 157–165, 1996.

[15] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, 2001.

[16] R. Poli, W. Langdon, N. McPhee, and J. Koza, *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008.