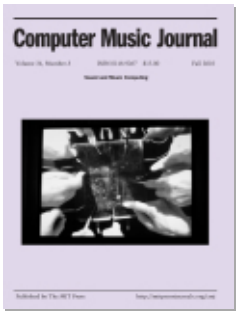
Machine Learning of Jazz Grammars

Jon Gillick, Kevin Tang, Robert M. Keller



Computer Music Journal, Volume 34, Number 3, Fall 2010, pp. 56-66 (Article)

Published by The MIT Press

Machine Learning of Jazz Grammars

Jon Gillick,* Kevin Tang,† and Robert M. Keller**

*Annkissam
One Broadway, 14th Floor
Cambridge, Massachusetts 02459, USA
jrgillick@wesleyan.edu
†Stanford University
Palo Alto, California 94305, USA
kdtang@stanford.edu
**Harvey Mudd College
301 Platt Blvd.
Claremont, California 91711, USA
keller@cs.hmc.edu

In the context of an educational software tool that can generate novel jazz solos using a probabilistic grammar (Keller 2007), this article describes the automated learning of such grammars. Learning takes place from a corpus of transcriptions, typically from a single performer, and our methods attempt to improvise solos representative of such a style. In order to capture idiomatic gestures of a specific soloist, we extend an earlier grammar representation (Keller and Morrison 2007) with a technique for representing melodic *contour*. Representative contours are extracted from a corpus using clustering, and sequencing among contours is done using Markov chains that are encoded into the grammar.

This article first defines the basic building blocks for contours of typical jazz solos, which we call *slopes*, then shows how these slopes may be incorporated into a grammar wherein the notes are chosen according to tonal categories relevant to jazz playing. We show that melodic contours can be accurately portrayed using slopes learned from a corpus. Experimental results, including blind comparisons of solos generated from grammars based on several corpora, are reported.

Related Work

Grammars form the basis of our melody generation technique (Keller and Morrison 2007). Use of grammars for creating musical structures has also been suggested or investigated by numerous researchers (Winograd 1968; Roads 1979; Bell and Kippen 1992; Cope 1992; McCormack 1996; Pachet 1999; Papadopoulos and Wiggins 1999; and others). Dubnov et al. (2003) used probabilistic and statistical machine learning methods for musical style recognition. Eck and Lapalme (2008) investigated automatic composition and improvisation with neural networks, and Cruz-Alcazar and Vidal-Ruiz (1998) developed a method for learning grammars to model musical style. The latter applied three grammatical inference algorithms to automatic composition of melodies in Gregorian, Bach, and Joplin styles, achieving the best results with the Gregorian melodies. They classified 20 percent of composed melodies as *very good*, which they defined as able to be "taken as an original piece from the current style without being a copy or containing evident fragments from samples." We strove toward the same definition of "very good" solos.

An important part of our representation involves a formalization of melodic contour. Kim et al. (2000) used contours for musical classification and querying, and Chang and Jiau (2003) investigated musical contour with applications to extracting repeating figures and themes from music. In addition, De Roure and Blackburn (2000) proposed melodic pitch contours for content-based navigation of music.

We use clustering as a means of organizing and abstracting a large variety of similar melodic fragments, and Markov chains to represent likely transitions between abstract fragments. Kang, Ku, and Kim (2001) used a graphical clustering algorithm for extraction of melodic themes. Jones (1981) described uses of both Markov chains and grammars in music composition. Verbeurgt, Dinolfo, and Fayer (2004), among others, used Markov models as a means for composition by learning transition

Computer Music Journal, 34:3, pp. 56–66, Fall 2010

probabilities between patterns. Ames (1989) dealt with different-sized Markov chains of notes.

Jazz Improvisation

Ideally, jazz improvisation involves the creation of new melodies while the melodies themselves are being performed. It is known that this process is often informed by prior creation and practice of vocabulary ideas prior to performance. One of the purposes of our work is to construct software tools that facilitate the construction and recording of such ideas. Such a tool can also be used to transcribe and analyze existent ideas. The present work shows that, once transcribed, a solo can be put to use in the creation of a grammar, which can then be used to provide improvisations over any chord progression, not just the ones in the corpus.

Although a given jazz performer might not be aware of how he or she does improvise, it seems reasonable to say that ideas of what one is able and willing to play can be captured in the form of patterns or, more generally, some form of grammar. It is obvious that a finite set of patterns can be described by an ad hoc grammar. A grammar that is too ad hoc, however, would tend to generate only very predictable, and thus eventually uninteresting, melodies. It is important, then, that melodic ideas be abstracted to enable the replacement of certain elements with others, so that novel results can be produced. If the abstraction is too coarse-grained, however, the melody may lose coherence.

Melody Generation Based on Abstraction

The purpose of jazz solo generation is to create novelty, while at the same time adhering to some structural and harmonic guidelines. Although melodic abstraction has obvious uses in analysis, we contend that it is very useful in generation as well. For example, one can generate a melody by first generating an abstraction of a melody, then *instantiating* that abstraction to an actual playable melody. We elaborate on this approach in the following.

Table 1. Note Categories Used in Grammar Terminals

Symbol	*Color*	*Meaning*
C	black	**Chord** tones of the current chord
L	green	**Color** tones (chord extensions)
H	——	Either a chord tone or a color tone ("**Helpful**" tones)
A	blue	Tones that chromatically **Approach** one of the above
——	red	Tones that are neither C, L, H, nor A
X	——	Arbitrary tone
R	——	Rest

Note Categories

Our grammatical approach for jazz melodies attempts to strike a balance between novelty and coherence by augmenting the note categories of Keller and Morrison (2007) that correspond to concepts in jazz playing. These categories are instantiated probabilistically and also in observance of other constraints, such as range considerations, at generation time. Each category, as given in Table 1, has a corresponding terminal symbol in the grammar and four of them show as different note head colors on the staff, for elucidation and feedback to the user. Owing to printing limitations, we do not show the colors in examples here, but will call out salient aspects of categories by annotation. One other category, S, for scale tones, was used by Keller and Morrison (2007), but it does not play a major role in the current work.

A grammar generates a list of *terminal symbols*, from which a melody is then generated. Each note corresponds to one terminal symbol, but the terminal symbol specifies only the note category, as described earlier, and the duration. The actual note retains the duration, but the pitch is generated probabilistically, and with certain attention being paid to the interval between it and the previous note. For example, in the terminal alphabet, C4 represents a chord tone of duration one quarter-note, L4/3 a color tone of a quarter-note triplet, A8 represents an approach tone of duration one eighth-note, H4 a dotted quarter-note chord or color tone, R2 a

Figure 1. Two example realizations of the S-expression (Δ 1 2 H8 H8 H8). The interpretation of this expression is an ascending group of three eighth notes that are chord or color tones, with each note separated in pitch from the previous by at least one semitone and at most two semitones.

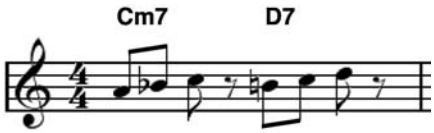
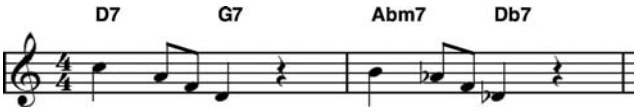


Figure 2. Two example realizations of the S-expression (Δ -3 -4 C4 H8 H8 C4). The interpretation of this expression is a descending series of a chord quarter note, two "helpful" eighth notes, and a chord quarter note, with a minimum separation of three semitones and a maximum separation of four semitones. ("Helpful" means either a chord tone or a color tone.)



half-note rest, etc. (By "color tones," we mean tones that are not in the chord, but that are complementary to it and have a sonorous quality. Sometimes these are referred to as "extensions" or "tensions" of the chord.) The symbols for note categories are derived from ones for expressing concrete melodies in Impro-Visor (Keller 2005). That notation was intended to make it easy for jazz musicians to enter and read melodies in textual form, and the category notation is aimed at achieving similar ergonomic goals.

Abstract Melodies

It is reasonable to regard a sequence of terminal symbols in the grammar as being an *abstract melody*, in the sense that multiple melodies will fit the sequence when the note categories are instantiated to corresponding pitches. Another advantage of such melodic abstractions is that they can be instantiated over any chord progression, even for chords of different quality, such as major, minor, diminished, dominant, etc.

Although individual note categories can be used to generate somewhat convincing jazz melodies, in order to capture specific styles it is necessary to introduce one or more mechanisms to provide greater coherence among individual notes. Thus we extend the individual note categories with "macros" that can capture sequences of notes in certain patterns. The current work focuses on a single macro concept, called a *slope*. Each slope has two numeric parameters, followed by a sequence of one or more terminal symbols. The numeric parameters indicate the minimum and maximum *rise* between successive notes in the sequence. Negative numbers indicate *fall* rather than rise. In the grammar rules, slopes are treated as terminal symbols appearing in the consequent of a production. In generating a melody, the terminal symbols inside a slope are converted to specific notes, as before.

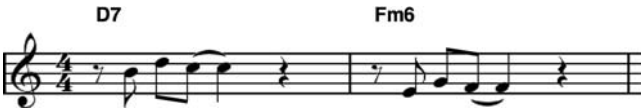
S-expressions (McCarthy 1960) are used in our grammar notation to provide grouping of notes in a sequence, and for hierarchy, when necessary. For brevity, we will represent the word "slope" by Δ in this paper. Figures 1–4 gives some examples of S-expressions for slopes, along with the interpretation of those expressions. It is not always possible to obey the constraints of both the slope and the note category, so sometimes the tool must give priority to one or the other.

Slopes are unidirectional, and thus not sufficient by themselves to represent all idioms of interest. For example, consider the bebop idiom of an *enclosure* (Baker 1998), wherein a chord tone is approached by notes above and below. It requires two slopes to represent this contour, as shown in Figure 3. Our notation for chords follows jazz lead sheet abbreviations, as given in Table 2.

In addition to short idioms, we can capture larger selections, such as in Figure 5, from Red Garland's solo on *Bye Bye Blackbird* (Davis 1961). Notes such as the G# in the first measure of the original melody in Figure 5 begin an ascending segment, and so have only one interval from which to choose a minimum and maximum slope. In such cases, we found that relaxing the step bounds by a half step in each direction yielded better results. Consequently, we relax (Δ-9-9 A16) to (Δ-8-10 A16) before instantiating to a melody.

Because chord tones play the most significant role in shaping the melody, we prioritize chord tones higher than slope bounds, but for note categories other than chord tones, we do not. Above the melody is a series of line segments intended to show the qualitative contour of the original line. The caption of Figure 5 shows the derived abstract melody. Figures 6 and 7 illustrate two melodies generated from that abstract melody. For contrast, Figure 8 shows what happens when we use only the note categories and ignore the contour information encoded as slopes, whereas Figure 9 shows using slopes, but suppressing note categories. In the last

Figure 3. Two example realizations of the S-expression (R8 L8 (Δ 3 5 H8) (Δ -2 -1 C4.)). The interpretation of this expression is a rest, followed by an eighth note color tone, followed by a chord or color tone three to five semitones up, then a chord tone one or two semitones down. This is an example of an "enclosure" (Baker 1998).



instance, more notes (shown circled) outside the harmony are generated.

If there is no tone of the given type available within the pitch bounds, we use a probability table to choose another note. This could involve relaxing the pitch bounds (usually when looking for chord tones) or selecting a note of a different type within the bounds (usually for all notes except chord tones). The purpose of such expedients is to avoid introducing a full-blown constraint-solving system into the implementation.

Grammar Learning

Grammatical inference algorithms attempt to define the rules of a grammar for an unknown language through analysis of a training data set. The data can contain both positive samples (strings in the language) and negative samples (strings that should not be accepted). In our case, we use only positive sample sets, e.g., performances from an artist that we want to attempt to imitate. The grammar, then, should generate strings corresponding to melodies similar to those in the training corpora, and ideally generate nothing that differs greatly from them.

We experimented with several methods for extracting grammar rules from training data, including extraction by phrases, with a phrase defined as a section of a solo starting after a rest and ending with a rest. Given the variable length of phrases and the difficulty of recombining phrases into a solo of a specified size, we settled on breaking melodies into *time windows* of a predefined length. After choosing two parameters, the number of beats per window and the number of beats by which to slide the window (which is necessarily less than or equal to the window size), we collect all melodic fragments of a certain length in a corpus and associate one grammar terminal for each abstract melody of the given length. We found that, among fragments of between 1 and 8 beats, 4-beat fragments achieved the best balance between originality and continuity in the case of 4/4 meter, which was the most typical.

Figure 4. An example realization of the S-expression: (R4 (Δ 5 5 C8/3 C8/3) (Δ -5 -5 C8/3) (Δ -2 -2 C8/3) (Δ 5 5 C8/3) (Δ -5 -5 C8/3) (Δ -1 -1 C8/3) (Δ 5 5 C8/3) (Δ -5 -5 C8/3) (Δ -2 -2 C4)) The interpretation of this expression is a standard jazz idiom, containing triplets descending diatonically by chord tones, with specified intervals.

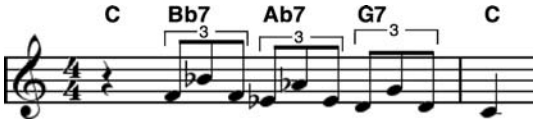


Table 2. Chord Symbols

Symbol	*Meaning*
M	major
m	minor
m7	minor seventh
7	dominant seventh, if by itself
6	added sixth

Markov Chaining

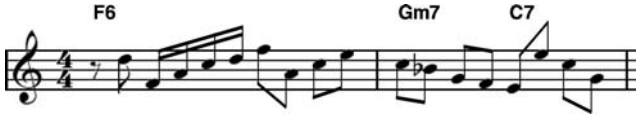
To improve continuity between melodic fragments, once we have gathered the abstract melodies that will make up our generated solos we combine them into full solos based on a *Markov chain* (Kemeny et al. 1959) represented within the grammar. Markov chains represent a system with a sequence of states, using conditional probabilities to model the transitions between successive states. Because our grammars are already probabilistic, Markov chains can be fit into the grammar framework more or less naturally. An n-*gram Markov* chain uses probabilities conditioned on the previous $n - 1$ states. Sets of abstract melodies serve as the states in the Markov chain. Given a starting melody, we add the next phrase based on a list of transition probabilities from the first measure.

Clustering

To increase the variety in recombination of melodic ideas, we *cluster* similar abstract melodies together using the k-*means clustering* algorithm (Hartigan and Wong 1979). We then collect statistics that represent which clusters follow other clusters in the corpus and build our table of probabilities accordingly, using clusters as states of the Markov chain. To compose new solos, we first generate a

*Figure 5. An original melody line, with contour suggested above it. An abstract melody (*slope *representation) is extracted from it as the S-expression: (R8 C8 (Δ -6 -6 A16)(Δ 1 3 C16 C16 C16 C8)(Δ -12 -12 C8) (Δ 2 4 C8 L8) (Δ -4 -1 L8 C8 C8 A8 C8)(Δ 12 12 C8) (Δ -12 -2 C8 C8)) Figures 6–9 show example realizations of this S-expression. The S-expressions are not always strictly adhered to (see text).*

Figure 6. A melody generated from the S-expression of Figure 5, using slopes.



sequence of clusters from the grammar, and then randomly select representatives from clusters, again using the grammar rules to specify the distribution of representatives.

Clustering algorithms represent data as points in an *n*-dimensional plane and group points together through some distance metric. Our cluster analysis is based on a Euclidean distance measure on seven parameters:

1. number of notes in the abstract melody,
2. location of the first note that starts within the window,
3. total duration of rests,
4. average maximum slope of ascending or descending groups of notes,
5. whether the window starts on or off the beat,
6. order of the contour (how many times it changes direction), and
7. consonance.

The *average maximum slope* (parameter 4 in the list) is obtained by segmenting the abstract melody into sequences of notes that are all ascending or all descending, calculating the slopes between each pair of neighboring notes within each sequence, taking the absolute value of each slope, finding the maximum absolute value within each sequence, and then averaging the maximum absolute values from all such sequences within the abstract melody.

The *consonance* value (parameter 7) is assigned to a measure based on the note categories. For each note, we add to the consonance value a coefficient for the note category multiplied by the duration of the note. For example, typical coefficients are 0.8 for a chord note, 0.6 for an approach note, 0.4 for a color note, and 0.1 for other notes.

Given a parameter *k* for the number of clusters, the *k*-means algorithm selects *k* points as cluster centers and then begins an iterative process given by the following two steps:

1. Assign each data point to the nearest cluster center.
2. Re-compute the new cluster centers.

These steps are repeated for some number of iterations or until few enough data points switch clusters between iterations. In principle, the user can set the value of *k*. However, we currently use for *k* the number of melodic fragments divided by 10 as a nominal empirical value. Figure 10 shows three representative one-measure melodies from a corpus of Charlie Parker solos. The algorithm first abstracted these melodies, and then clustered the corresponding abstract melodies together.

Transition Probabilities

Transition probabilities of the Markov chain are based on clusters as states. In the training data, real melodies are mapped to abstract melodies, which are then mapped to clusters, as described previously. Then we re-consult the training data and count, for each pair of clusters *A*, *B* how many times a melody from cluster *B* follows a melody from cluster *A*. Normalizing these counts gives us the transition probabilities between clusters. Figure 11 illustrates the result for clusters derived from a small corpus.

Grammar Representation

We encode, into the grammar, the transition probabilities as derived from clustering. In effect, the transition probabilities generate a sequence of clusters, and then an abstract melody is chosen from each cluster and instantiated to a real melody.

Table 3 shows a simple probabilistic grammar corresponding to the transitions in Figure 11. Note that each non-terminal symbol has an argument indicating the number of measures to be filled. This is only used to control the expansion of the

Figure 7. Another melody generated from the S-expression of Figure 5, using slopes.

Figure 8. A melody generated from the S-expression of Figure 5, using only note categories, and not slopes.

Figure 9. A melody generated from the S-expression of Figure 5, using only slopes, and not note categories. Circled notes are neither chord nor color tones.

Figure 10. Three melodies instantiated from representatives within one cluster of a Charlie Parker solo corpus.

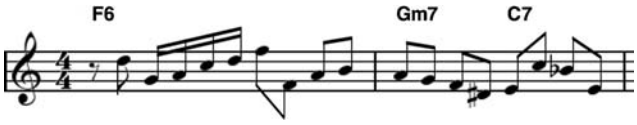


Figure 7.

Figure 8.

Figure 9.

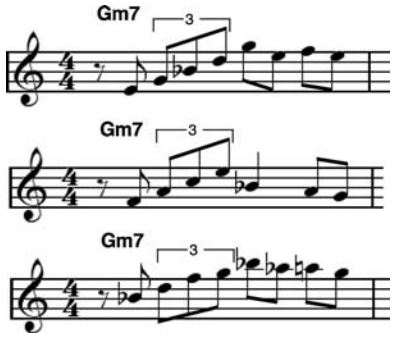


Figure 10.

grammar to fill a certain amount of temporal space, the number of measures of which is specified as the argument to the Start production. Non-terminals Start, C0, C1, and C2 represent the states of the Markov chain. The rules with Q0, Q1, Q2 on the right-hand sides show the choices from clusters. For brevity, we have included only one of the representative abstract melodies for each of Q0, Q1, and Q2. We have transcribed our implementation's notation from S-expressions to more conventional grammar rules for readability.

The grammar of Table 3 is for a first-order Markov chain. For an nth-order chain, where $n \geq 2$, the states would be labeled by sequences of n-1 cluster indices. These correspond to sequences of clusters that can actually occur in the corpus.

Generating with a Grammar

The terminal string derived from the grammar is an abstract melody, created by chaining together a representative abstract melody in each cluster. The representatives are currently chosen by a uniform distribution. We then generate a real melody by randomly selecting an initial note of the specified note category and then filling in the rest from slope and note category constraints.

Algorithm Summary

The following summarizes the algorithmic approach used in the current work, beginning with the analysis phase, which creates a grammar from a corpus of solos over chord progressions, followed by synthesis, which creates new solos over possibly quite-different chord progressions. There will be many syntheses performed for a given analysis phase, as our users will mostly be using grammars synthesized by themselves or others.

Analysis, resulting in a grammar:

1. Break up the corpus of transcriptions into melodic fragments (time windows), typically one measure in length.
2. Translate each fragment into an abstract melody made up of slopes, note categories, and rhythms.
3. Run a clustering algorithm on abstract melodies, grouping all abstract melodies into clusters, with typically ten abstract melodies per cluster on average.
4. Reexamine each transcription in the corpus to determine the sequence in which clusters appeared.
5. Collect n-gram statistics on the clusters, typically for n between 2 and 4, at the option of the user.

Figure 11. Example of derived transition probabilities between clusters.

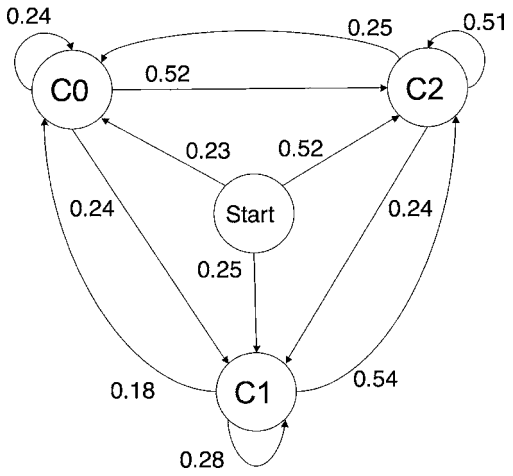


Table 3. Probabilistic Grammar Embedding a Markov Chain

Production Rule	*Probability*
Start(Z) → C0(Z)	0.23
Start(Z) → C1(Z)	0.25
Start(Z) → C2(Z)	0.52
C0(0) → ()	1
C1(0) → ()	1
C2(0) → ()	1
C0(Z) → Q0 C0(Z-1)	0.24
C0(Z) → Q0 C1(Z-1)	0.24
C0(Z) → Q0 C2(Z-1)	0.52
C1(Z) → Q1 C0(Z-1)	0.18
C1(Z) → Q1 C1(Z-1)	0.28
C1(Z) → Q1 C2(Z-1)	0.54
C2(Z) → Q2 C0(Z-1)	0.25
C2(Z) → Q2 C1(Z-1)	0.24
C2(Z) → Q2 C2(Z-1)	0.51
Q0 → ((Δ 0 0 R2 R4 R8 C16/3) (Δ 1 1 A16/3 L16/3)	1
Q1 → ((Δ 0 0 C8) (Δ -9 -9 C8) (Δ 2 3 C8 G4+8 R4))	1
Q2 → ((Δ 0 0 C4/3) (Δ 1 2 L4/3 A4/3) (Δ -7 -1 C4/3 G4 C8/3))	1

6. Create a probabilistic grammar that generates sequences of clusters based on the *n*-gram data.

Melody synthesis, using a grammar:

1. Given a leadsheet with a chord progression but no melody, the user selects a section of the tune over which to generate a solo.
2. The grammar rules expand, using production probabilities, until a sequence of clusters that corresponds to the desired number of measures is generated.
3. From each cluster in the sequence, randomly choose one abstract melody, and concatenate the abstract melodies.
4. Translate the abstract melodies into music by probabilistically selecting notes that obey the constraints of slope and note category as designated by the abstract melody.
5. (Optional): At the user's discretion, the generated melody can be rectified automatically, pulling any notes not classified as chord, color, or approach into line with the harmony by a half-step correction. This corrects for any conflicts between the specified abstract note qualities and slope specifications.

Qualitative Results

Figures 12 and 13 show two solos created using our approach, both over the same chord progression, a 12-bar blues. The grammars were learned from the corpora of Charlie Parker and John Coltrane, respectively. Nine solos (*Anthropology*, *Cheryl*, *Dewey Square*, *Laird Baird*, *Moose the Mooche*, *Now's the Time*, *Ornithology*, *Scrapple from the Apple*, and *Yardbird Suite*) were used for learning from the Parker corpus, and two solos (*Giant Steps* and *Moment's Notice*) for the Coltrane corpus. Both musicians were known to create relatively dense solos. In contrast, Figure 14 shows a solo generated from a grammar learned from a Miles Davis corpus of one solo (*On Green Dolphin Street*). Davis was known for much less density, and this clearly shows in the line generated by the grammar. However, we should reemphasize that the differentiating characteristics of the generated solos are based entirely on the transcribed solos that were learned. Given a different corpus, say from a different point in the soloist's career, noticeably different results are apt to be generated.

Figure 12. Twelve-bar blues solo generated using Charlie Parker–style grammar.

Figure 13. Twelve-bar blues solo generated using John Coltrane–style grammar.

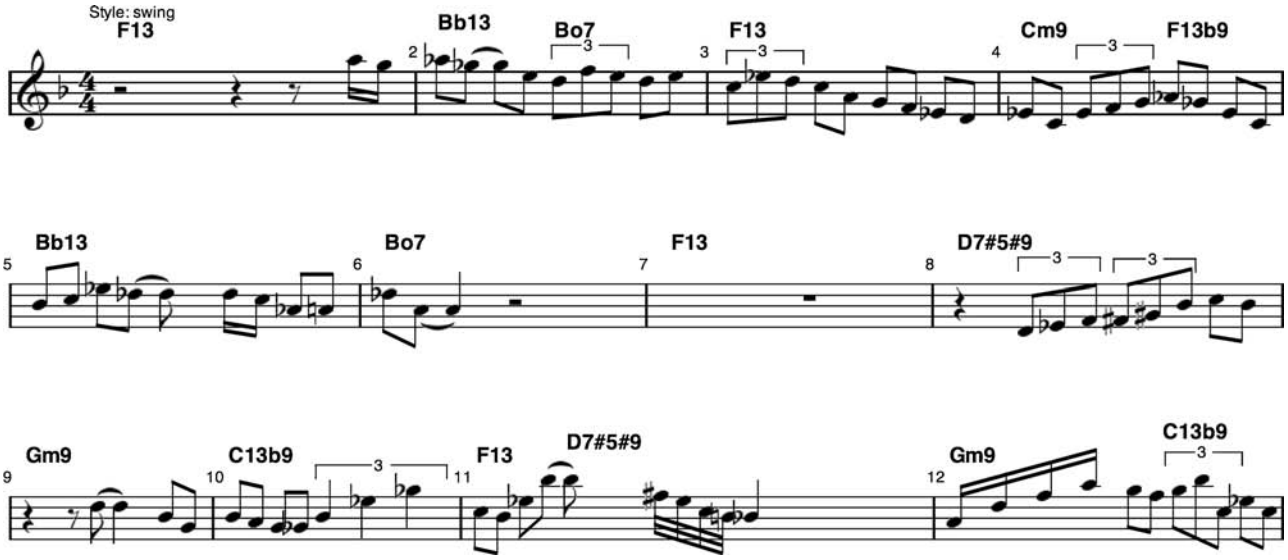


Figure 12.

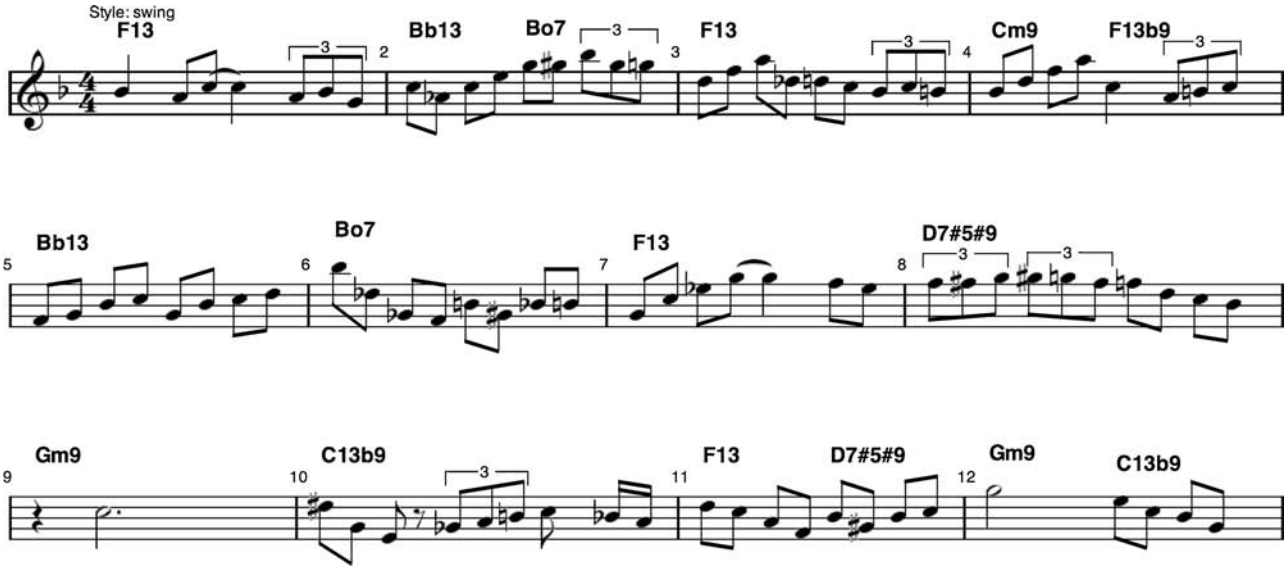
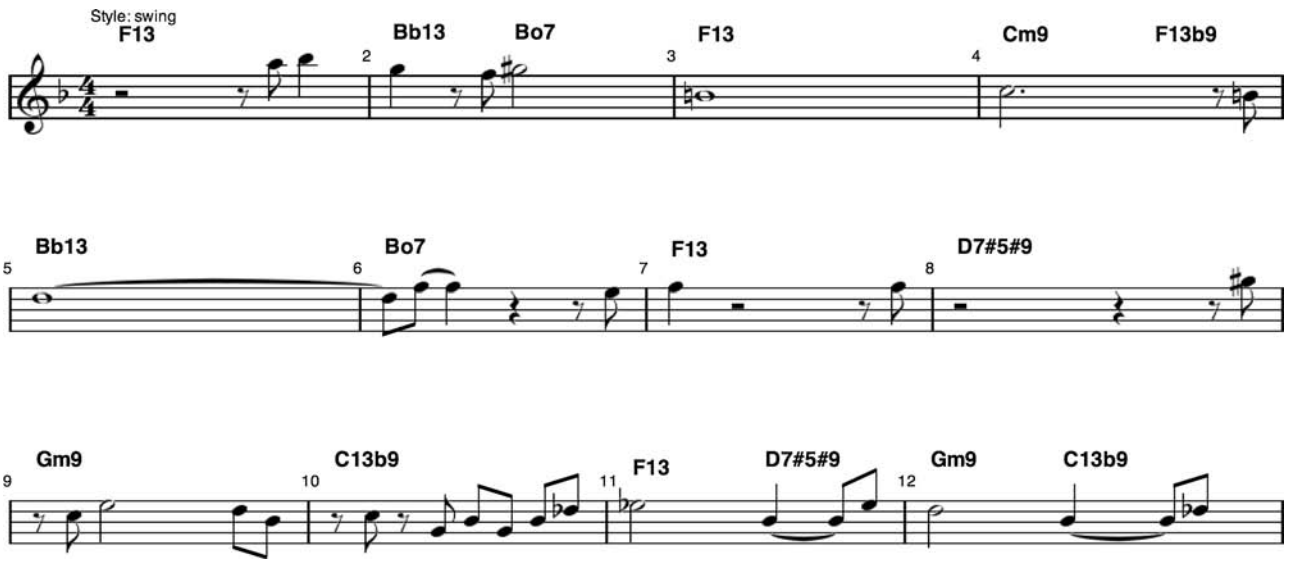


Figure 13.

Implementation and Usage

Our ideas and methods were added as a learning extension of the solo-generation functionality of the open-source software tool Impro-Visor (Keller 2007), implemented in Java, for which both the executable and source code are publicly available. To learn a grammar in Impro-Visor, the user makes available one or more solo transcriptions that have been encoded in Impro-Visor's leadsheet format (Keller 2005). These are collected together in a common directory. Then a few button presses will generate

Figure 14. Twelve-bar blues solo generated using Miles Davis–style grammar.



a grammar. The following parameters are currently settable by the user:

1. Window size, in beats
2. Window slide amount, in beats
3. Number of representatives per cluster
4. Order of the Markov chain

Qualitative Evaluation

For short solos of four to eight measures, we found that our algorithm's compositions usually sound like a reasonably capable jazz soloist and occasionally like a convincing imitation of an artist. Longer solos were more apt to lack a sense of direction. Four-gram Markov chains produced longer coherent passages than bi-gram and tri-gram models, and were also able to generate very good 12- or 16-measure solos about 25 percent of the time. Higher order *n*-grams, for $n > 4$, seemed to provide no more coherence than the four-gram case. However, with a much larger training set than our largest, which was around 400 measures of Charlie Parker solos, it is possible that higher order *n*-gram models could regularly yield coherent solos of 16 or more measures.

Blind Evaluation by Third Parties

To measure our method's effectiveness at style emulation, we set up an experiment to determine whether or not test subjects could match the styles of three prominent jazz trumpet players with solos composed in the style of each player. We inferred grammars for Clifford Brown, Miles Davis, and Freddie Hubbard from 72 bars of solos from each. Each subject listened to one transcribed solo from each artist and one generated solo from each grammar, with each generated solo being over the chord progression for *Bye Bye Blackbird*, six solos total. Without revealing the names of the artists, we asked the subjects to try to match the emulated artist from each computer-composed solo with the actual artist from the transcribed solo. We also asked subjects to qualitatively indicate how close the resemblance was by "Not close," "Somewhat close," "Quite close," or "Remarkably close."

Out of 20 test subjects, 95 percent correctly matched the computer-composed solo in Clifford Brown's style to the original Clifford Brown solo, 90 percent correctly matched Miles Davis, and 85 percent correctly matched Freddie Hubbard. Of the same subjects, 85 percent correctly matched all three solos. All but one undecided subject characterized

the resemblance to the original artists as either "Somewhat close" or "Quite close," with 45 percent "Somewhat close" and 50 percent "Quite close." On a scale of 1 to 10, half of the subjects rated their own musical knowledge between 2 and 5, and half between 6 and 9.

Conclusion and Future Work

The ability of our method to generate solos that sound similar to the artist from the training data, yet distinct from any particular solo, shows that our method of data abstraction is effective. The combination of contours and note categories seems to balance similarity and novelty sufficiently well to be characterized as jazz. In addition, clustering appears to be a workable algorithm for grouping similar abstract melodic fragments. Markov chains were effective in structuring solos; however, additional global structure is desirable for providing intra-solo coherence.

A more convincing test of our method would be an experiment to determine whether listeners, particularly jazz musicians, can tell the difference between a solo generated by a learned grammar and a human-composed solo.

There is much that can be done at the grammar level to permit greater expressiveness and succinctness. We introduced slopes to represent contours, but it is clear that the general idea of macro constructs other than slopes could prove very useful. Examining more specific information about notes in conjunction with the note categories, such as interval from the root of a chord, could also prove beneficial.

The greatest weakness of our generated long solos is their lack of global structure. A more conclusive evaluation of the effectiveness of *n*-grams for global structure could be done given a sufficiently large training set. Another approach that could be explored is to construct the high-level structure of a generated solo based on a particular solo from the training set.

Acknowledgments

This work was supported by grant 0753306 from the National Science Foundation. We thank the anonymous referees for providing input on earlier drafts.

References

Ames, C. 1989. "The Markov Process as a Compositional Model: A Survey and Tutorial." *Leonardo* 22(2):175–187.

Baker, D. 1998. *How To Play Bebop*, I. Van Nuys, California: Alfred.

Bell, B., and J. Kippen (1992). "Bol Processor Grammars." In M. Balaban, K. Ebcioglu, and O. Laske, eds. *Understanding Music with AI: Perspectives on Music Cognition.* Cambridge, Massachusetts: MIT Press, pp. 366–400.

Chang, C.-W., and H. C. Jiau. 2003. "Extracting Significant Repeating Figures in Music by Using Quantized Melody Contour." *Proceedings of the Eighth IEEE International Symposium on Computers and Communication*. New York: IEEE, pp. 1162–1167.

Cope, D. 1992. "Computer Modeling of Musical Intelligence in EMI." *Computer Music Journal* 16(2):69–83.

Cruz-Alcazar, P., and E. Vidal-Ruiz. 1998. "Learning Regular Grammars to Model Musical Style: Comparing Different Coding Schemes." *Proceedings of the Fourth International Conference on Grammatical Inference.* New York: Springer-Verlag, pp. 211–222.

De Roure, D. C., and S. G. Blackburn. 2000. "Content-Based Navigation of Music Using Melodic Pitch Contours." *Multimedia Systems* 8(3):1–11.

Dubnov, S., et al. 2003. "Using Machine-Learning Methods for Musical Style Modeling." *Computer* 36(10):73–80.

Davis, M. 1961. *Miles Davis in Person, Friday Night at the Blackhawk, San Francisco.* Audio recording (LP). Columbia C2K 87097.

Eck, D., and J. Lapalme. 2008. "Learning Musical Structure Directly from Sequences of Music." Technical report 1300. Montreal: Université de Montréal, Département d'informatique et de recherche opérationnelle.

Hartigan, J. A., and M. A. Wong. 1979. "Algorithm AS 136: A k-Means Clustering Algorithm." *Applied Statistics* 28(1):100–108.

Jones, K. 1981. "Compositional Applications of Stochastic Processes." *Computer Music Journal* 5(2):45–61.

Kang, Y.-K., K.-I Ku, and Y.-S. Kim. 2001. "Extracting Theme Melodies by Using a Graphical Clustering Algorithm for Content-Based Music Information Retrieval." *Proceedings of the Fifth East European Conference on Advances in Databases and Information Systems*. New York: Springer-Verlag, pp. 84–97.

Keller, R. M. 2005. "Impro-Visor Leadsheet Notation." Available from www.cs.hmc.edu/~keller/jazz/improvisor/LeadsheetNotation.pdf. Last accessed 26 April 2010.
Keller, R. M. 2007. Impro-Visor, Jazz Improvisation Advisor, www.cs.hmc.edu/~keller/jazz/improvisor/. Last accessed 26 April 2010.
Keller, R. M., and D. R. Morrison. 2007. "A Grammatical Approach to Automatic Improvisation." *Proceedings of the Fourth Sound and Music Computing Conference*. Athens: National and Kapodistrian University of Athens, pp. 330–337. Available online at http://smc07.uoa.gr/SMC07%20Proceedings.htm.
Kemeny, J. G., H. Mirkil, J. L. Snell, and G. L. Thompson. 1959. *Finite Mathematical Structures.* 1st ed. Englewood Cliffs, New Jersey: Prentice-Hall.
Kim, Y. E., et al. 2000. "Analysis of a Contour-Based Representation for Melody." *Proceedings of the International Symposium on Music Information Retrieval.* Amherst, New York: University of Massachusetts. Available online at http://ciir.cs.umass.edu/music2000/indexnoframes.html.
McCarthy, J. 1960. "Recursive Functions of Symbolic Expressions and their Computation by Machine." *Communications of the ACM* 3(1):184–195.
McCormack. 1996. "Grammar-Based Music Composition." In R. Stocker et al., eds. *Complex Systems 96: From Local Interactions to Global Phenomena.* Amsterdam: IOS Press, pp. 321–336.
Pachet, F. 1999. "Surprising Harmonies." *International Journal on Computing Anticipatory Systems* 4:1–20.
Papadopoulos, G., and G. Wiggins. 1999. "AI Methods for Algorithmic Composition: A Survey, a Critical View and Future Prospects." *AISB Symposium on Musical Creativity.* Menlo Park, California: American Association for Artificial Intelligence, pp. 110–117.
Roads, C. 1979. "Grammars as Representations for Music." *Computer Music Journal* 3(1):48–55.
Verbeurgt, K., et al. 2004. "Extracting Patterns in Music for Composition via Markov Chains."*Proceedings of the 17th International Conference on Innovations in Applied Artificial Intelligence*. New York: Springer-Verlag, pp. 1123–1132.
Winograd, T. 1968. "Linguistics and Computer Analysis of Tonal Harmony." *Journal of Music Theory* 12(1):2–49.