

ImprovGenerator: Online Grammatical Induction for On-the-Fly Improvisation Accompaniment

Kris M. Kitani
University of Electro-Communications
1-5-1 Chofugaoka, Chofu
Tokyo, Japan
kitani@is.uec.ac.jp

Hideki Koike
University of Electro-Communications
1-5-1 Chofugaoka, Chofu
Tokyo, Japan
koike@is.uec.ac.jp

ABSTRACT

We propose an online generative algorithm to enhance musical expression via intelligent improvisation accompaniment. Our framework called the *ImprovGenerator*, takes a live stream of percussion patterns and generates an improvised accompaniment track in real-time to stimulate new expressions in the improvisation. We use a mixture model to generate an accompaniment pattern, that takes into account both the hierarchical temporal structure of the live input patterns and the current musical context of the performance. The hierarchical structure is represented as a stochastic context-free grammar, which is used to generate accompaniment patterns based on the history of temporal patterns. We use a transition probability model to augment the grammar generated pattern to take into account the current context of the performance. In our experiments we show how basic beat patterns performed by a percussionist on a cajon can be used to automatically generate on-the-fly improvisation accompaniment for live performance.

Keywords

Machine Improvisation, Grammatical Induction, Stochastic Context-Free Grammars, Algorithmic Composition

1. INTRODUCTION

The ultimate goal of this work is to improve the interaction between a musician and a computer interface to stimulate more expressive and innovative performance. In this work, we focus on a type of extemporaneous musical expression between two percussionists (e.g. street drumming, sheds). The goal is to simulate the interaction of the second performer that reacts to the musical nuances and suggestions of the other. This type of human-machine feedback is sometime referred to as computational synergetics [15] or machine improvisation [2].

Current devices available on the market for improvisation accompaniment are highly deterministic and have minimal capacity for real-time human-computer interaction. Drum machines provide basic drum loops and provide some limited control over the temporal structure. They can work in

conjunction with triggers to insert drums fills and breaks on-the-fly. A delay effect is also commonly used in live performances to provide interaction between a performer's current musical input and a repeated sample from a previous duration. Loop samplers also provide a framework for generating complex combinations of sounds as a rhythmic base track for improvisation accompaniment. These devices however are limited by the fact that they can only produce what has been programmed by the user ahead of time.

2. RELATED RESEARCH

In comparison to the accompaniment devices available on the market, work with algorithmic composition has developed more sophisticated frameworks for generating accompaniment or improvisation tracks. The *Voyager* system created by Lewis [6] used a set of predefined subroutines to generate accompaniment and solos. Keller [3] used grammars to automatically generate improvisation for jazz melodies by using a rich set of abstractions. McCormack [8] used a L-system grammar and Markov chains for stochastic musical composition. Kippen and Bel [4] used a context-free grammars to represent the structure of Indian drumming. However, a common characteristic of a majority of past approaches is that they require an expert knowledge engineer to design the rules of the generative model. This process of designing the grammar is in itself an art.

In order to simplify the learning process, Thom implemented an unsupervised learning algorithm [12] and used the model to generate real-time improvisation patterns. The learning phase, however, requires an offline learning process, that learns parameters from a predefined improvisation data set. The *Omax[1]* system used a factor oracle model to both learn and generate rhythmic, harmonic and melodic patterns in real-time. The interactive jamming system by Weinberg [14] used a Markov chain controlled by a high-level state machine to learn and generate patterns for a multi-player percussion performance. These models are well suited for real-time learning and generative improvisation but only explicitly encode the flat (non-hierarchical) structure of the input patterns. Hierarchies can be useful for understanding a performance at multiple temporal granularities and can also be used to generate patterns over various temporal windows.

In contrast to previous work, our system learns a hierarchical set of grammar rules online without supervision and these rules can immediately be used to generate improvisation accompaniment. In particular our *ImprovGenerator* system uses a probabilistic mixture model to generate accompaniments track based on the (1) current musical context (i.e. what is being played now) and the (2) long-term

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

NIME2010, June 15-18, 2010, Sydney, Australia

Copyright 2010, Copyright remains with the author(s).

pattern motifs (i.e. what has been played in the past). Since our system learn a grammar online, there is no need to prepare a train dataset.

3. CONTEXT-FREE GRAMMAR

To model the long-term patterns encoded in the live performance, we utilize a grammar as the generative model. Although grammars were originally developed for problems in computational linguistics, grammars have also been applied to a wide range of problems such as RNA modeling [11], plan recognition [10], human activity modeling [5] and music analysis [7]. In this work we use a context-free grammar (CFG) to model rhythm patterns because of their ability to explicitly and compactly describe hierarchical structure.

Formally, a context-free grammar is defined by the 4-tuple $\mathbf{G} = \{\mathbf{T}, \mathbf{N}, S, \mathbf{R}\}$, where \mathbf{T} is a finite set of terminal symbols, \mathbf{N} is a finite set of non-terminal symbols, S is the start symbol (a special non-terminal symbol) and \mathbf{R} is the set of production rules. The production rules take the form $A \rightarrow \lambda^*$, which states that non-terminal symbol A produces the string λ^* of one or more symbols. When a probability $P(A \rightarrow \lambda^*)$ that satisfies the condition $\sum_i P(A \rightarrow \lambda_i^*) = 1$, is associated to each rule, the grammar becomes a stochastic context-free grammar (SCFG).

When a SCFG is used for rhythm patterns, each terminal symbol represents an attack and each non-terminal symbol represents an abstraction of a substring of terminal symbols (e.g. some combination of notes). The start symbol S represents a single time window (e.g. a bar), a complete symbol string produced by the grammar. From a more qualitative perspective the SCFG encodes information about a performer's playing style. By analyzing the grammar, one can extract both common and rare phrases within a musician's performance.

4. LEARN AND GENERATE

4.1 Online grammar learning

We implement a heuristic online CFG learning algorithm called Sequitur [9] to learn the structure of a real-time stream of beat patterns. The Sequitur algorithm works by enforcing two conditions on the input symbol pattern. The *bi-gram uniqueness* conditions stipulates that all bi-gram (a sequence of two symbols) in the grammar be unique. This first condition creates new rules for redundancies in the input string. The second condition called the *rule utility* ensures that every rule is used at least once. This condition has the effect of remove rules that are used only once as a result of the learning process. The details of the algorithm can be found in the original paper [9].

In this paper we experiment with a form percussion improvisation that is built upon a certain base pattern (e.g. afro beat, samba). The role of the CFG is to capture the long term base pattern. For our experiments, we heuristically insert a bar count (a numeric terminal symbol) as a unique symbol at the head of each bar to limit our learning to within individual bars (i.e. not across bars). The time window can easily be adjusted to handle patterns that span several bars (e.g. 3-2 clave pattern). With a one bar time window the learned SCFG can be used to probabilistically generate accompaniment after two bars have been observed. At this point, all patterns produced by the generative grammar are copies of patterns observed from the user. However, the probabilities assigned to the grammar rules ensure that reoccurring phrases in the live performance are also frequently generated by the grammar.



Figure 1: Hardware setup

4.2 Mixture model

The accompaniment patterns generated by the grammar are only repetitions of previously observed patterns and therefore do not take into account the current context of the performance (e.g. what is the other percussionist doing now?). To make the system respond to both the history of the performance and the current context, we use a mixture model that produces the final accompaniment pattern by taking both contexts into account.

Each one bar pattern \mathbf{x} (e.g. sixteen 16th notes) is drawn from a probability distribution conditional on all past observations $\mathbf{y}_{1:t}$, which is decomposed into a distribution represented by the grammar \mathbf{G} and a conditional transition probability that depends on the previous observation \mathbf{y}_{t-1} , where α is the weighting parameter of the two distributions.

$$\mathbf{x} \sim p(\mathbf{x}_t | \mathbf{y}_{1:t}) \quad (1)$$

$$= \alpha p(\mathbf{x}_t | \mathbf{G}) + (1 - \alpha) p(\mathbf{x}_t | \mathbf{y}_{t-1}) \quad (2)$$

In our current implementation the transition probability is computed simply from the previous observation \mathbf{y}_{t-1} (i.e. the probability for each note is a zero or one). The transition probabilities can also be computed from the statistics over a larger time window.

This mixture model gives the system the ability to take into account both the entire history of the performance $1 : t$ and also the current context of the performance at time t .

5. SYSTEM OVERVIEW

The prototype system (Figure 1) consists of four components: (1) Cajon, (2) USB microphone, (3) speaker-amp and (4) laptop computer.

5.1 ImprovGenerator

Chuck [13] is used to process the feature extraction and execute the timed audio feedback. The communication between the Chuck module and grammar induction model is facilitated with the open sound control (OSC) protocol. Rhythm patterns are transmitted to the induction module at the end of each 4 beat increment and the SCFG is updated in real-time. Once the grammar induction module has processed at least one bar of rhythm patterns, the pattern generator is able to send an accompaniment pattern to the Chuck module in real-time to trigger a set of stored audio files (depicted in Figure 2). For simplicity the current system uses a click track to give the musician a time reference but can be replaced with a beat tracker for more flexibility as in [14].

5.2 Feature extraction

To detect the onsets of the Cajon, we implement a simple threshold over the output energy (volume), the rollover frequency (85%) and the RMS of the frequency spectrum

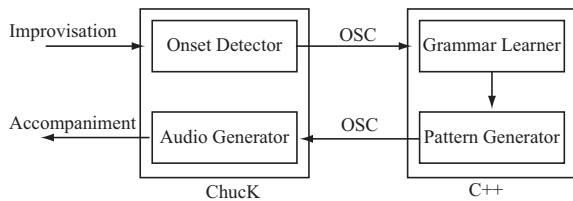


Figure 2: System flow chart

Table 1: Learned Grammar: Brazilian Samba

Production Rule		Probability	
S	→	B	(0.065)
S	→	D	(0.161)
S	→	O	0001 1001 (0.032)
S	→	G	(0.065)
S	→	0010 M 0111	(0.032)
S	→	J	(0.226)
S	→	K 0001	(0.032)
S	→	N	(0.226)
S	→	1010 1001 H	(0.032)
S	→	1000 L	(0.032)
S	→	M C	(0.032)
S	→	1010 1010 H	(0.032)
S	→	O C	(0.032)
A	→	0000 0000	(1.000)
B	→	A A	(1.000)
C	→	1001 1001	(1.000)
D	→	C C	(1.000)
G	→	1010 1000 H	(1.000)
H	→	1101 0111	(1.000)
J	→	1010 L	(1.000)
K	→	C 1001	(1.000)
L	→	1011 H	(1.000)
M	→	1011 1001	(1.000)
N	→	K 1001	(1.000)
O	→	1001 1000	(1.000)

over a small time window. We found that this thresholding was sufficient to detect onsets and still remain robust to nominal environmental noise. We note here that it is also possible to use MIDI messages or trigger responses as the input stream. This framework is not dependent on the mode of the detector.

6. EXPERIMENTAL RESULTS

Two short improvisation sequences were performed to test the ImprovGenerator: (1) Brazilian Samba and (2) 4/4 Afro beat. Each performance consisted of about 30 bars, beginning with the basic pattern leading into a series of improvised variations. Table 1 shows the grammar learned for the Samba sequence. The two sentential productions (i.e. production rules that begin with $S \rightarrow$) marked in boldface encode the two basic Samba patterns used throughout the performance. The other rules are variants of the basic patterns. The terminal symbols are denoted as four bit binary strings, where the bit represents a 16th note (e.g. 1010 is two eighth notes). The non-terminal symbols are denoted using capitalized alphabet symbols. The probability of each production rule is given in the right-side column and have been calculated using the maximum likelihood estimates from the observation counts.

Similar results are shown for the 4/4 Afro beat (Table 2). In contrast to the Samba sequence a single sentential production (also shown in boldface) accounts for a third of the probability distribution, which encodes the base pattern of the Afro beat. It is noted here that the grammars are very deterministic, such that after a sentential production has been drawn from the probability distribution, the resulting

Table 2: Learned grammar: Afro beat

Production Rule		Probability	
S	→	B	(0.059)
S	→	E	(0.353)
S	→	M 1001	(0.029)
S	→	N	(0.088)
S	→	L 1000 1000	(0.029)
S	→	K 0010	(0.029)
S	→	Q	(0.059)
S	→	O	(0.059)
S	→	0001 1010 1010 1001	(0.029)
S	→	M 1010	(0.029)
S	→	K 1011	(0.029)
S	→	1 0010 P	(0.029)
S	→	U	(0.059)
S	→	1101 1110 R	(0.029)
S	→	T	(0.059)
S	→	1101 0100 R	(0.029)
A	→	0000 0000	(1.000)
B	→	A A	(1.000)
E	→	K 1000	(1.000)
J	→	1001 1010	(1.000)
K	→	J 1010	(1.000)
L	→	1001 0010	(1.000)
M	→	J 0010	(1.000)
N	→	K 1001	(1.000)
O	→	K 1010	(1.000)
P	→	1010 1000	(1.000)
Q	→	L P	(1.000)
R	→	0010 1000	(1.000)
T	→	1001 0110 P	(1.000)
U	→	M 1000	(1.000)

pattern of the grammar is always the same. Next, it is shown how these grammatical patterns can be augmented by the use of a probabilistic mixture model.

An excerpt from both sequences are given in Figure 4 and Figure 3 to show how the mixture model (Eq. (2)) effects the final output sequence. Each excerpt shows the rhythmic input (top), the pattern generated by the grammar (middle) and the final pattern generated by the mixture model. For these example, the weighting parameter α was set to 0.8 giving more weight to the pattern generated by the grammar. The subtraction and addition of notes are marked by arrows to show how the performance context (i.e. the last bar performed by the musician \mathbf{y}_{t-1}) effects the final output. Since the weight parameter was set to be biased toward the grammar production, the performance context has only a slight effect on the final output. When the weight parameter is set to 0 the algorithm ignores the grammar production and simply repeats the last bar performed by the percussionist.

7. DISCUSSION

The mixture model used in this paper is simply a proof of concept and can be modified to take into account a wider range of temporal relationships within an improvisation performance by using higher order Markov models and allowing the grammar induction to span multiple bars.

The feature set and quantization of the current system obviously have space for improvement. Currently our system only detects attacks as the input feature and therefore future work should take into account other features such as accents, muted attacks, flams, etc. The quantization of notes (currently 16th notes) can be improved to recognize other variations like triplets and 32nd notes by increasing the granularity of the quantizer, combined with the use of a higher quality onset detector.

Although the grammars learned from online induction are not as powerful as those grammars that can be learned offline with supervision, this type of unsupervised approach is

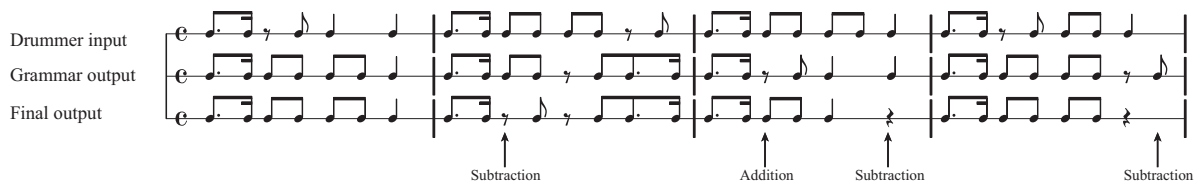


Figure 3: Accompaniment generated for the 4/4 Afro beat pattern. Top: Drummers input pattern, Middle: Grammar generated pattern, Bottom: Final output pattern

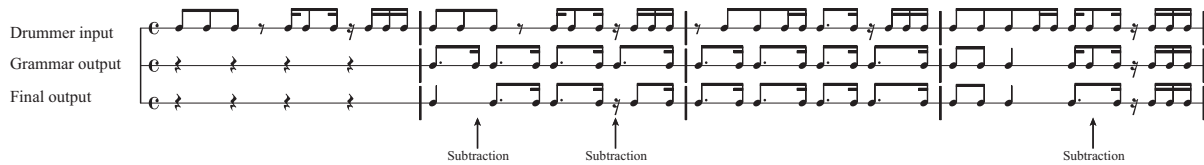


Figure 4: Accompaniment generated for the Samba pattern. Top: Drummers input pattern, Middle: Grammar generated pattern, Bottom: Final output pattern

needed when the structure of the performance is unknown and training sets are not given a priori. The current implementation also does not take full advantage of the generative capabilities of a SCFG (e.g. cyclic rules, multiple production rules). Learning other grammars in parallel may help to discover more meaningful production rules.

8. CONCLUSIONS

We have presented a fully automated system for online learning and real-time generation of improvisation accompaniment. By generating accompaniment patterns based on both the history of input patterns and the current context of the performance via the mixture model, the ImprovGenerator provides a prototype framework for enhancing improvisation performances. Our preliminary experimental results show that our framework is able to capture both the general thematic texture of the performance via the hierarchical grammar while also having the flexibility to respond to the expressions of the current performance.

9. REFERENCES

- [1] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov. OMax brothers: a dynamic topology of agents for improvisation learning. In *Proceedings of the 1st ACM workshop on Audio and music computing multimedia*, page 132. ACM, 2006.
- [2] G. Assayag and S. Dubnov. Using factor oracles for machine improvisation. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9):604–610, 2004.
- [3] R. M. Keller and D. R. Morrison. A grammatical approach to automatic improvisation. In *Proceedings of the Sound and Music Computing Conference*, pages 330–337, 2007.
- [4] J. Kippen and B. Bel. Modeling music with grammars: Formal language representation in the Bol processor. In *Computer Representations and Models in Music*, pages 207–238, 1988.
- [5] K. M. Kitani, Y. Sato, and A. Sugimoto. Recovering the basic structure of human activities from noisy video-based symbol strings. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(8):1621–1646, 2008.
- [6] G. Lewis. Too many notes: Computers, complexity and culture in voyager. *Leonardo Music Journal*, pages 33–39, 2000.
- [7] P. Mavromatis and M. Brown. Parsing context-free grammars for music: A computational model of Schenkerian analysis. In *Proceedings of the International Conference on Music Perception and Cognition*, pages 414–415, 2004.
- [8] J. McCormack. Grammar based music composition. *Complex Systems 96: From Local Interactions to Global Phenomena*, 3:320–336, 1996.
- [9] C. G. Nevill-Manning and I. H. Witten. Identifying hierarchical structure in sequences: A linear-time algorithm. *Journal of Artificial Intelligence Research*, 7:67, 1997.
- [10] D. V. Pynadath and M. P. Wellman. Generalized queries on probabilistic context-free grammars. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1285–1290, 1996.
- [11] Y. Sakakibara, M. Brown, R. C. Underwood, I. S. Mian, and D. Haussler. Stochastic context-free grammars for modeling RNA, 1993.
- [12] B. Thom. Learning models for interactive melodic improvisation. In *Proceedings of the International Computer Music Conference*, pages 190–193, 1999.
- [13] G. Wang and P. Cook. ChucK: A programming language for on-the-fly, real-time audio synthesis and multimedia. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 812–815, New York, NY, USA, 2004. ACM.
- [14] G. Weinberg, A. Raman, and T. Mallikarjuna. Interactive jamming with Shimon: a social robotic musician. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 233–234. ACM, 2009.
- [15] N. J. Zabusky. Computational synergetics: visualization and vortex dynamics. *Journal of Computational and Applied Mathematics*, 22(2-3):285–295, 1988.