Control Improvisation with Applications to Music

Sophie Libkind, Alexandre Donzé, Sanjit Seshia, David Wessel
UC Berkeley

August 2, 2013

Abstract

A French politician once said that "Music is too serious a matter to be left to musicians"[1]. Without trying to fully justify this intentionaly provocative statement, we would argue that indeed music theory has known deep connections with mathematics, computer science(s), embedded systems, cyber-physical systems, etc, on which the active field of computer music is built. And as such, much has to be gained from experts in those fields to drive their attention to music, in order to bring their expertise to musicians, but also to benefit from a fun and extremely rich domain of experimentation and research for their own work. This work, which is concerned with automatic music generation was initiated in this spirit.

The more general setting for our work is that of *control improvisation.* In a controlled cyber-physical system, a controller provides a stream of control inputs to the plant. This stream of inputs, in a discretized setting, can be viewed as a string of letters drawn from a specified alphabet. (For example, the alphabet can correspond to a set of events.) In many settings, such control sequences can be computed at design time, given a specification of the plant and other parts of the environment. However, when the environment does not behave as specified (e.g., in a disaster situation), the controller must "improvise" so that, at a minimum, certain critical safety properties are satisfied. Moreover, the sequence of control inputs should ideally not deviate "too much" from the original sequence. The problem of improvising a control sequence to satisfy safety constraints while deviating little from the original sequence is termed as the *control improvisation* problem. We describe here our work on this problem in the context of generating music.

Music generation can be done either at the audio or at a symbolic level. The former involves processing and synthesizing sound waves eventually produced by some speaker(s), whereas the later is concerned only with generating scores, i.e., sequences of (groups) of symbols, the notes, each of them being an abstract representation of a particular sound that can be instantiated in many different variations by different instruments or, generally speaking, by sound synthesizers. Thus, at the symbolic level, the problem can be reduced to that of generating sequences of letters, each of which corresponding to a note. Improvisation is obtained from a training sequence by identifying recurrent patterns, imitating and recombining them. Different data structures and algorithms have been proposed for this purpose such as incremental parsing (IP) [6] inspired from dictionary based compression algorithms such as those from the Lempel-Ziv family [2], probabilistic suffix trees (PST) [5] and more recently factor oracles (FO) [1].

In this work, we consider the scenario of solo generation over a given jazz song harmonization. In this case, the improvised sequence has to be synchronized with another sequence, usually chords progressions, considered as fixed and called thereafter the accompaniment. The improviser then has to be a function of both the training sequence and the chord sequence. [8] introduces Impro-visor, a system of automatic jazz improvisation that uses a stochastic context free grammar (SCFC) which can be learned [7], to create licks which are concatenated to form longer jazz pieces. We take a different approach, based on factor oracles. The approach we propose can be decomposed into two components: one *generator* that implements an improviser for simple melodies, as described above and ignoring a priori the accompaniment and one *supervisor* which

[1]George Clemenceau (1841-1929) - although the authors do not guarantee the exactitude of the wording, quoted from memory and translated from French.

imposes rules to the generator so that it plays in harmony with the accompaniment. The rules are similar to "safety properties" that a control system must always obey. Both are implemented using compositions of finite state machines in a scheme inspired from the fied of supervisory control of discrete-event systems [3].

An important question in music generation is that of the similarity between the generated material and existing melodies. Many measures have been proposed often in the purpose of genre classifications. Among these measures, Normalized Compression Distances (NCDs) are considered as a universal classifiers and have been used in this context of styles clustering of musics [4]. Here we implement a variant of an NCD to estimate the divergence between generated melodies and the training melody. As it is based on a dictionary based compression algorithm which can also be used for the purpose of sequence generation, we observe that for this particular measure of divergence, it is then possible to derive (or synthesize) an improviser with a precise control over the divergence.

We implemented and tested a first prototype of our approach in Python and current work aim at integrating it within more appropriate frameworks, for instance Ptolemy [9] for a modular and flexible implementation of the different heterogenous components of our improvisers, and Max [10] for experiments in live and realtime.

References

[1] G. Assayag and S. Dubnov. Using factor oracles for machine improvisation. *Soft Comput.*, 8(9):604–610, 2004.

[2] G. Assayag, S. Dubnov, and O. Delerue. Guessing the composers mind: Applying universal prediction to musical style. In *Proceedings of the International Computer Music Conference*, pages 496–499, 1999.

[3] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[4] R. Cilibrasi, P. Vitanyi, and R. De Wolf. Algorithmic clustering of music. In *Web Delivering of Music, 2004. WEDELMUSIC 2004. Proceedings of the Fourth International Conference on*, pages 110–117. IEEE, 2004.

[5] S. Dubnov, G. Assayag, and O. L. G. Bejerano. A system for computer music generation by learning and improvisation in a particular style. *IEEE Computer*, 10(38), 2003.

[6] S. Dubnov, G. Assayag, and R. El-Yaniv. Universal classification applied to musical sequences. In *Proceedings of the International Computer Music Conference*, pages 332–340, 1998.

[7] J. Gillick, K. Tang, and R. M. Keller. Learning jazz grammars. In *Proceedings of the SMC 2009 - 6th Sound and Music Computing Conference*, Porto, Portugal, 2009.

[8] R. M. Keller and D. R. Morrison. A grammatical approach to automatic improvisation. In *Proceedings SMC'07, 4th Sound and Music Computing Conference*, pages 330 – 337, Lefkada, Greece, 2007.

[9] `http://ptolemy.eecs.berkeley.edu`. Ptolemy II.

[10] `http://www.cycling74.com`. Max/msp/jitter version 6.1.