Learning Harmonic Progression Using Markov Models

EECS545 Project
Bradley J. Clement
bradc@umich.edu
April 25, 1998

1. Introduction

In an effort to try to automate the learning of the structure of musical compositions, we recognize that music, like natural language, has a "deep structure" that is difficult to represent. One barrier lies in our ability to even understand the semantics of music. The computer music society is interested in the generation and representation of music, and there are preliminary investigations in learning music that serve as motivation for this project.

Here a method and implementation of learning probabilistic grammars is described. A wide range of problems, such as speech recognition and biological sequence modeling, can benefit from probabilistic language modeling, and here we apply a simple method to musical style representation and the generation of music. This method is a simple approach falling into a class of Bayesian model-merging techniques investigated by Andreas Stolcke. The basic idea is to create a completely specific representation of example data sequences and generalize the model to predict/generate new instances. The example data of interest here are harmonic progressions. These can be used to describe one dimension of the musical style of a composer. However, other aspects of music could also be learned (to the extent representable given language restrictions) using this method: pitch, duration, tempo, dynamics, melody, phrasing, and rhythm. Assuming that these properties can be learned independently, the real challenge is to learn how they depend on each other to give a full representation of musical style.

In section 2, previous work on grammar learning and their application to the domain of music are described. In section 3, a simple technique based on the methods described in section 2 is explained, and experimental results are given in section 4.

2. Background

2.1. Music Representation and Learning

There has long been interests in representing music mainly from the point of view of developing tools to assist composers. The experience of artificial intelligence researchers has shown problems imposing formalization on natural language; however, searching for representations for low-level musical components, such as pitch or duration, has led to the well-accepted MIDI format and many tools used by composers. Most of the representations seen in music and AI literature use generative rules or grammars. These grammars can be categorized by their expressivity as regular, context-free, or context-

sensitive. In many cases, probabilistic information is incorporated into the generative rules.

2.1.1. Grammar representations

Regular grammars can be described as state machines or finite automata (FA). Each state can be thought of as a decision point where a part of the final sequence of data (music in the context of this paper) is determined. State transitions determine how these pieces are ordered with respect to each other. Although cycles in the state machine offer some degree of modular structure, it is not able to describe many intricate multi-level tree-structures that are evident in high-level composition. For example, the string of n *a*'s followed by n *b*'s, $a^n b^n$, cannot be represented in a finite number of states. For this reason, many believe that regular languages are not adequate for describing music. In general, this seems obvious, but it may be the case that they can represent lower-level characteristics of music, such as harmonic progressions discussed in this work. This would be good news since the computational aspects of handling these structures are relatively simple and well-known.

Context-free grammars are not limited in the ways described in conjunction with regular grammars and have been successful in representing higher-level constructs of composition. Where the rewrite rules for regular grammars translate a state into a terminal symbol followed by the next state, context-free grammars translate a state into any sequence of terminals and states allowing for higher-level state definitions (non-terminals), which provide greater modularity and brevity of description. However, context-free languages are limited in that in different contexts of the string formation the rules are always the same. Context-sensitive grammars allow generative rules to be defined in terms of high or low-level states occurring before and after the present one. Dealing with these grammars is computationally complex, but one method to alleviate this complexity is to add control procedures that can be used to track the context of the grammar and adjust (or select) an appropriate lower-level context-free representation [Roads, 1979].

2.1.2. Applications of learning to music

Since there is a large body of research applying learning techniques to language, it is not surprising that they have already been applied to music. Gerhard Widmer developed a system that learns rules for two-voice counterpoint composition by asking a human expert questions and applying explanation-based learning techniques [Widmer, 1992]. Belinda Thom uses incremental learning methods for an online system that to try to predict the next chord in a sequence of jazz music [Thom, 1995]. Recurrent neural networks have also been used to learn patterns in music. [Burr et.al., 1993][Mozer, 1994] Another approach to music learning by David Cope served as a motivation for the work presented here [Cope, 1992].

Cope has developed a process in which he does pattern-matching on the pitch and duration intervals of notes of a piece to recognize *signatures* of the composer. He then uses a hand written augmented transition network (ATN) to generate harmonic progressions according to rules for tonal music. An ATN is a hierarchy of finite state

machines where a state in one finite state machine can represent another finite state machine. He then maps the signatures of the piece onto the chords generated by the ATN to create a new piece that tries to reflect the style of the composer. However, this somewhat arbitrary generation of harmonic progression does not take into account any aspects of the music upon which he applies pattern-matching. The aim of this project in conjunction with a previous class project is to extend Cope's work by learning the harmonic progressions of the composer.

2.2. Learning Techniques

Several techniques have been applied to grammar learning, some of which are noted in the previous section as applied to the domain of music. The problem investigated here differs from others in that only positive example data are given. This means we do not consider examples of music that do not conform to the style of the composer. This also means that the target model is not available for active query. In other words, we cannot ask the composer if a particular piece of music conforms to his style. Simplifications where active query is possible allows us to use polynomial time algorithms for learning a regular language. All other cases of learning the smallest consistent finite automaton (FA) are NP-hard [Freund et. al., 1993]. In addition, non-deterministic finite automata (NFA) and context-free grammars cannot be identified in polynomial time if even given access to the target model. [de la Higuera, 1997]

2.2.1. Hidden variables

The experiments presented in this project are based on a Markov model representation. A Markov model is simply a NFA with probabilities associated with each state transition. Each state is assigned a single output symbol or set of symbols, or, in other terms, it has a single emission. This differs from a hidden Markov model (HMM) where at any state all emissions have some probability. So, the simpler Markov model is a special case of a HMM where one emission has a probability of 1 while all others have 0 probability. Thus, HMMs have greater expressivity in terms of modeling underlying *hidden* processes or variables that are not accessible.

For example, a sequence of coin toss results could be explained in a number of ways. In Figure 1a, we show a model of the process of generating coin tosses in terms of a Markov model. In one state (or node) the results is *heads* while *tails* occurs in the other state. State transitions between the states (arcs) represent a fair coin toss where both states are equally likely. Figure 1b is an equivalent HMM where now each state has two emissions, one for *heads* and one for *tails*. Although the probabilities of the emissions differ between the states, the probabilities of getting *heads* or *tails* in one step is the same as Figure 1a regardless of the current state. This model can be described as having a hidden underlying process at each state that determines the result given that state. So, a fair coin is used to determine the state transition as in Figure 1a, but biased coins are used to determine the result for each state. Rabiner and Juang assert that the models in Figure 1a and 1b are distinguishable, but the author of this report does not agree. However, the model in Figure 1c is clearly distinguishable from the other two in that the probability of the result

depends on the current state and the probabilities assigned to the arcs between states [Rabiner and Juang, 1986].

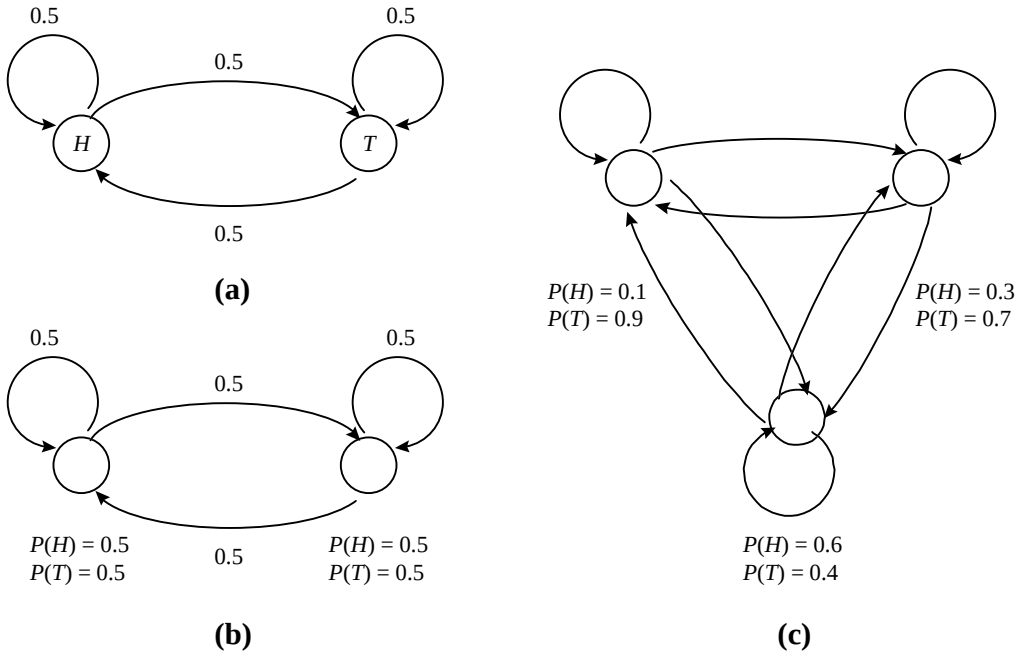


Figure 1. Hidden Markov Models

Context-free grammars can use probabilistic information in a similar way. These are known as stochastic context-free grammars (SCFGs). We can assign probabilities to transformations described by the rewrite rules. Without loss of generality, we can assume that any nonterminal can be rewritten as any other non-terminal with some probability. This allows us to represent hidden processes in SCFGs in a way similar to HMMs [Stolcke, 1994].

2.2.2. Bayesian approaches

In searching for the best model for a set of example data, whether it is a model of a regular or context-free grammar, Bayesian techniques are often used to evaluate the model. Rabiner and Juang present algorithms for determining the probability of the example data set (X) given an HMM (M), or $P(X|M)$, using the forward-backward algorithm and for adjusting the transmission and emission probabilities of a model to maximize $P(X|M)$ using the Baum-Welch expectation maximization algorithm [Rabiner and Juang, 1986]. However, maximizing $P(X|M)$ for an example data set would prefer a completely specific Markov model where disjoint chains for each example datum are included. Therefore, something is needed to drive generalization of the model. This can be accomplished by noting that

$$P(M|X) = \frac{P(X|M)P(M)}{P(X)}$$

according to Bayes Rule. We can use the concept of Occam's Razor to prefer generalized models by defining $P(M)$, the probability of the model, in terms of its description length. This can be done in many ways using the number of states, transmissions, and emissions. The level of generalization can be controlled by a parameter, λ, that intuitively represents the number of occurrences of the example data [Stolcke, 1994]. Noting that the probability of the data, $P(X)$, is a constant, the equation we now wish to maximize becomes

$$P(M|X) = P(X|M)^{1/\lambda} P(M) \quad (1).$$

With this tool for evaluation, we now look at search algorithms for finding the best model.

2.2.3. Bayesian model merging

While a wide variety of techniques have been applied to grammar learning in general, the techniques for adapting a grammar model are not as common. One simple technique involves building a Markov model iteratively by walking along the path of observed symbols for each example datum and creating new states where none existed. Counts on arcs are incremented at each time they are traversed to record probabilistic transitions. This method is susceptible to overgeneralization since it only considers one example in a merge decision [Ron, 1995].

Other techniques start with the most specific Markov model (disjoint chains from start to finish) for all data and then generalize. A simple approach is to look at all pairs of states and see if the k preceding nodes for each have the same emissions. If so, the nodes can be merged. This is done iteratively until no more nodes can be merged. k is altered to control the generalization.

The Bayesian model merging approach involving HMMs starts with the specific model and looks at all possible pairs of states to merge. It then chooses the merge operation that would have the highest probability according to Equation 1, using the Baum-Welch algorithm to adjust probabilities for expectation maximization. It does this iteratively until no merge will increase $P(M|X)$. This hill-climbing approach can be easily adapted to a beam search to trade solution quality for computation time. Note that these methods are suboptimal—no tractable optimal methods are known for NP-hard problems such as this. An example of how this algorithm learns a Markov model for a^+b^+ from two examples, *ab* and *abab*.is shown in Figure 2. The initial model M_0 is the set of disjoint chains (excluding the initial and final states) for the specific examples. Note that the first two merge operations do not change the log likelihood of the data given the model; since *ab* begins every string, the model is not changed equivalent ignoring size. The model then generalizes by merging states into M_4, the model we expect (given our sample set). The log likelihood decreases, but $P(M|X)$ increases because $P(M)$ is now greater.

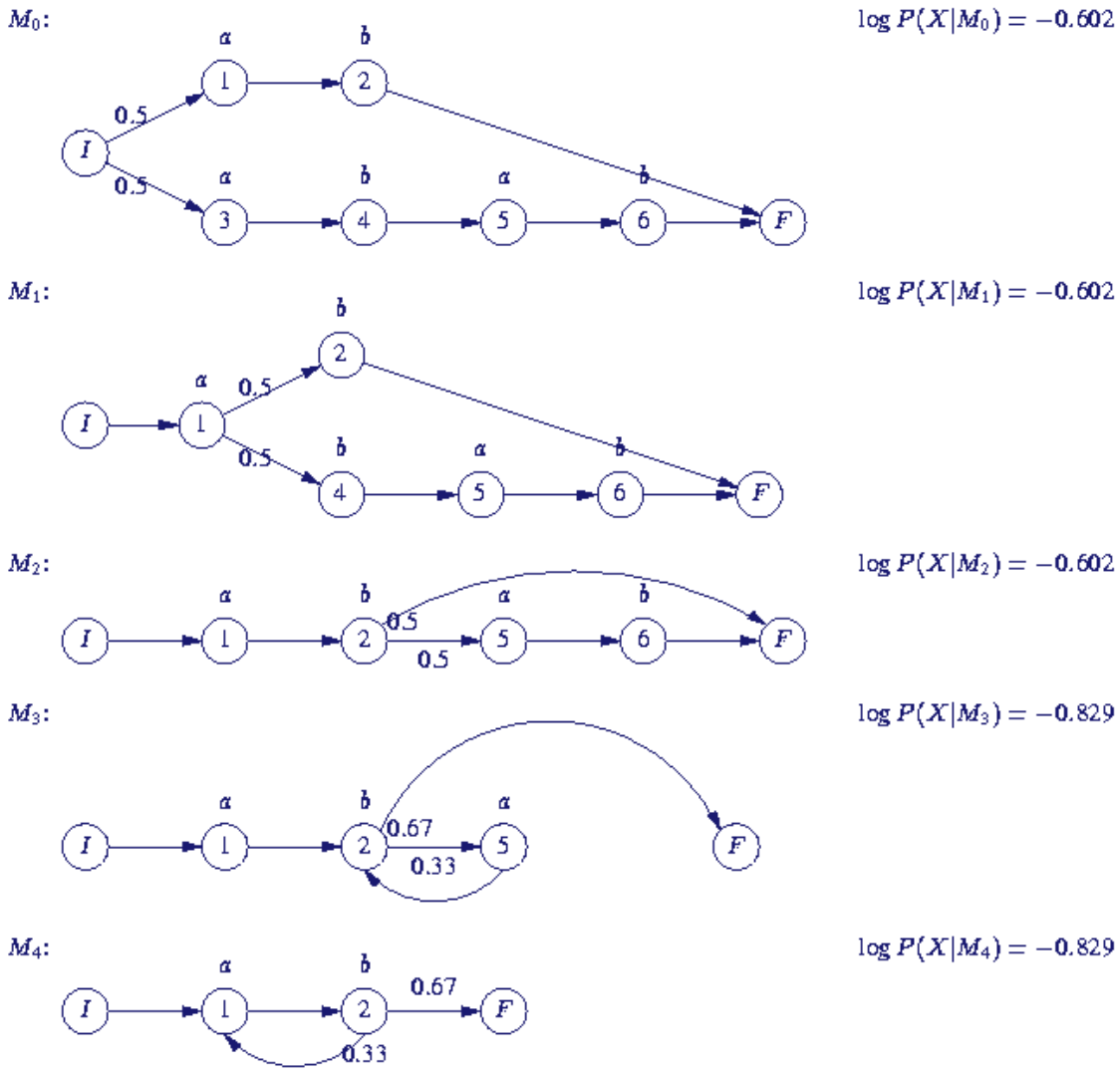


Figure 2. Markov model merging

SCFGs can also be generalized, but the method is somewhat different. There are two basic operations, *merge* and *chunk*. In the productions shown below, μ and λ are strings of terminals and non-terminals, and c is a count of the number of times the production fires to generate the sample data. Merging takes two non-terminals on the right-hand sides from different productions and combines them into a single non-terminal:

$$X_1 \Rightarrow \mu_1\,(c_1)$$
$$X_2 \Rightarrow \mu_2\,(c_2)$$
$$merge(X_1, X_2) = Y$$
$$Y \Rightarrow \mu_1\,(c_1)\,|\,\mu_2\,(c_2)$$

Chunking is replacing a substring of non-terminals on the right-hand side with a new non-terminal:

$$Z \Rightarrow \mu X_1 X_2 .. X_k \lambda \ (c)$$
$$Chunk(X_1 X_2 .. X_k) = Y$$
$$Z \Rightarrow \mu Y \lambda \ (c)$$
$$Y \Rightarrow X_1 X_2 .. X_k \ \ (c)$$

The operations are different than those applied to Markov models, but the higher-level algorithm is basically the same. We look at all possible merge/chunk operations and pick the one that will best improve $P(M|X)$ with the help of an EM algorithm to adjust probabilities. This is then repeated until a local optimum is reached.

3. Approach

The experiment reported here uses the Bayesian model merging method for Markov models as illustrated in Figure 2. Compared to HMMs, Markov models allow us to use simpler algorithms for calculating $P(X|M)$ and expectation maximization since emission probabilities are no longer considered. The forward-backward algorithm for calculating $P(x|M)$ is simplified to summing over all distinct paths generating the example datum x the products of the transition probabilities. Then we simply calculate

$$P(X|M) = \frac{\sum_{x \in X} P(x|M)}{|X|}.$$

Expectation maximization is simplified by a relaxation assumption that there is only a single path through the model for an example datum. By keeping track of counts of the number of data that share a path through transition (summing counts on newly shared transitions when merging nodes), we can update the counts for the most likely paths (Viterbi paths) of the data through the model [Stolcke, 1994][Rabiner and Juang, 1986].

The data used in the experiment uses a simple representation for harmonic progressions as input data. This data was actually generated from a program written by Andrew Nierman that outputs a number of phrases made up of symbols representing chord classifications according to rules of tonal music. In addition, parameters can be varied to create a style within tonal music constraints. This allows us to learn models for different styles and compare them by comparing probabilities of new data for classification.

4. Experiments

After implementing the approach described in section 3, the following experiments were performed to evaluate the learning method:

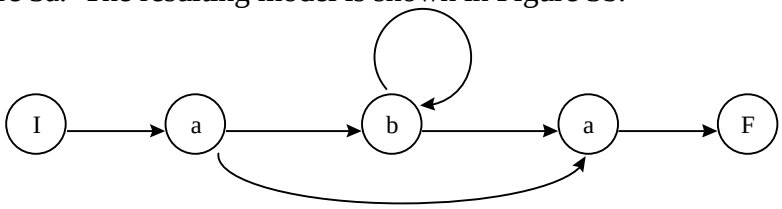
- Sanity checks on the learning algorithm were performed on simple regular languages:
  - ab^*a
  - a^*b^*
  - $a^+b^+a^+b^+$

- For $ac^*a \cup bc^*c$, λ (from Equation 1) was varied to show how it can control generalization.
- An evaluation of learning harmonic music was performed through learning the grammars of two different styles of tonal music.

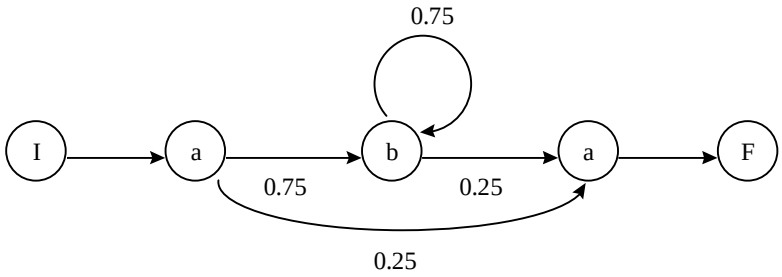
The results of each of the experiments are detailed in the following subsections.

4.1 Learning $\mathbf{ab^*a}$

4 examples of ab^*a were used to successfully learn the target Markov model shown in Figure 3a. The resulting model is shown in Figure 3b.



(a)

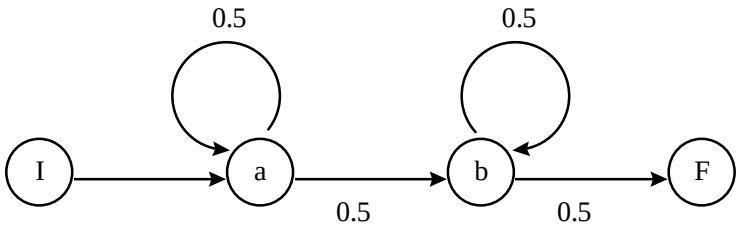


(b)

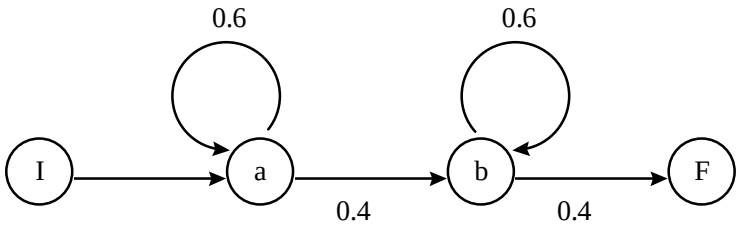
Figure 3. ab^*a

4.2. Learning $\mathbf{a^*b^*}$

The algorithm successfully learned the target in Figure 4a. The result is shown in Figure 4b.



(a)

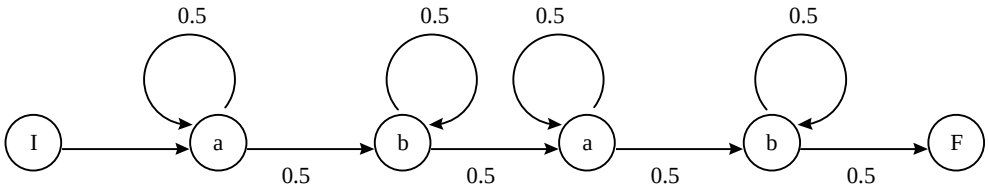


(b)

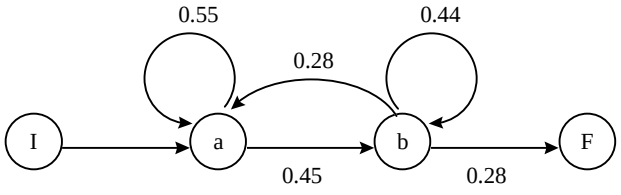
Figure 4. a^*b^*

4.3. Learning $\mathbf{a^+b^+a^+b^+}$

13 examples of $a^+b^+a^+b^+$ were used to try to learn the target Markov model shown in Figure 5a. The longest input was 11 characters. The result shown in Figure 3b illustrates that the algorithm failed to generalize correctly. As λ was varied, it was seen that the algorithm would over-generalize in the first a^+b^+ states before generalizing the second a^+b^+ states. Using the full Baum-Welch algorithm or beam search may have resulted in better generalization, but the work of Stolcke also shows this example being problematic in those cases.



(a)



(b)

Figure 5. $a^+b^+a^+b^+$

4.4. Learning $ac^*a \cup bc^*c$

In this experiment, the effects of varying λ were investigated. As shown in Figure 6a, when λ is chosen too small, the algorithm is less likely to merge states and generalize. Figure 6b shows the effects of choosing λ too large. In this case, the algorithm overgeneralizes. Selecting the optimal λ gives the correct model in Figure 6c.

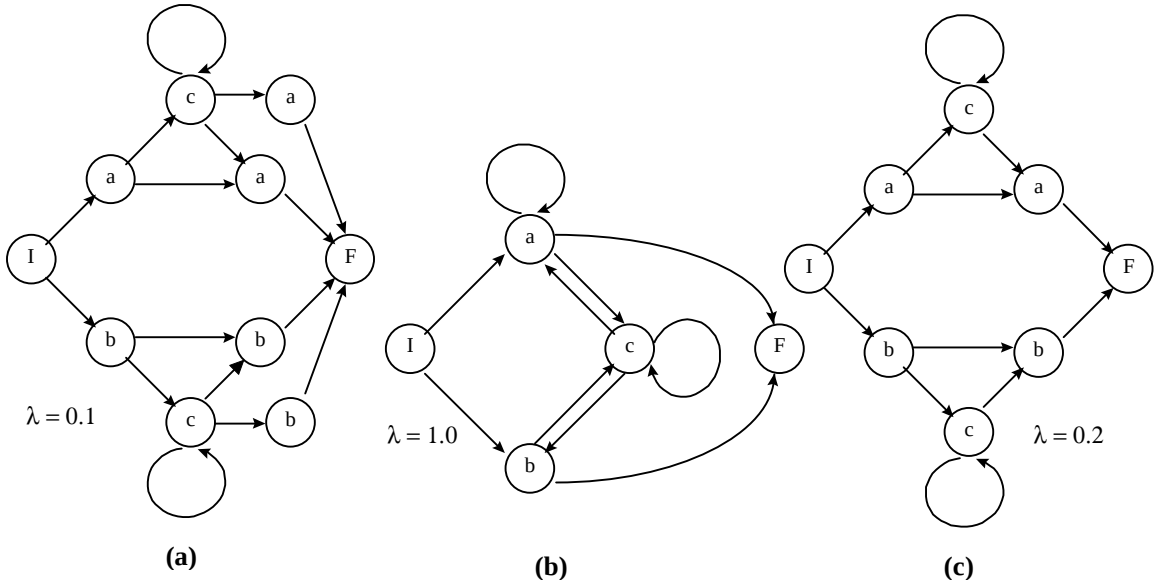


Figure 6. ***ac*a ∪ bc*c***

4.4. Learning harmonic progression

Here a program was used to generate two styles of harmonic progressions in chord sequences falling within the rules of tonal music. For each style, 54 example phrases were generated and learned. The styles were varied by altering the probabilities that certain chords will follow the others. Below are list descriptions of the two styles. The numbers in parentheses represent the relative likelihood of a chord to follow the first chord in the list on each line. For example, in the first line of the first description, IV or V is twice as likely to follow I as VI and six times more likely than II and III.

Harmonic Progression Style 1	**Harmonic Progression Style 2**
(I ((IV 6) (V 6) (VI 3) II III)) (II ((V 6) (IV 3) (VI 3) I III)) (III ((VI 6) (IV 3) I II V)) (IV ((V 6) (I 3) (II 3) III VI)) (V ((I 6) (IV 3) (VI 3) II III VII)) (VI ((II 6) (V 6) (III 3) (IV 3) I)) (VII ((III 6) (I 3)))	(I ((IV 8) (V 3) (VI 2) II III)) (II ((V 6) (IV 3) (VI 3) I III)) (III ((VI 4) (VII 8) (IV 4) I II V)) (IV ((V 3) (I 3) (II 6) III VI)) (V ((I 7) (IV 3) (VI 2) II III VII)) (VI ((II 6) (V 4) (III 5) (IV 2) I)) (VII ((III 9) (VI 2) IV))

The following descriptions represent the learned models for each of the styles. By fully merging nodes, it was easy to choose a large λ to achieve the desired effect. The relative probabilities are roughly the same, and with more examples should converge on the target styles. Of course, these descriptions make the assumption that style in the form of harmonic progressions can be represented as state transitions with associated probabilities,

but this is based on prior work that concludes that only a low-level of expressiveness is needed to represent tonal music in terms of harmonic progressions.

Learned Progression Style 1	Learned Progression Style 2
(I ((VI 11) (IV 16) (V 25) (II 2) (III 3))) (II ((IV 7) (V 9) (I 2) (VI 2))) (III ((VI 7) (IV 3) (II 1) (V 1))) (IV ((II 5) (I 6) (III 1) (V 18) (VI 2))) (V ((IV 13) (VI 9) (II 4) (III 2) (I 17) VII)) (VI ((III 5) (II 8) (V 11) (I 4) (IV 3))) (VII ((III 1)))	(I ((V 15) (IV 28) (III 6) (VI 4) (II 7))) (II ((IV 2) (V 13) (VI 7) (III 2) (I 3))) (III ((VII 7) (VI 5) (V 3) (IV 4))) (IV ((II 11) (V 6) (I 8) (VI 1))) (V ((IV 9) (I 15) (II 2) VII III VI)) (VI ((II 7) (V 5) (III 5) (IV 3))) (VII ((VI 2) (III 5) (IV 1)))

5. Conclusion

While there are apparent drawbacks in the ability of the investigated algorithms using Markov models to learn a great variety of languages (seemingly even regular languages), harmonic progressions are believed to be simple enough for the algorithms to perform well. When considering other aspects of music, Markov models may not be expressive enough, but it is hoped that algorithms for learning context-free grammars will be successful.

There is much future work in comparing the ability of HMMs and SCFGs to learn harmonic progressions as well as the many other aspects of music. Integrating the learning for the different characteristics appears to be a challenging task.

References

D.J. Burr, Y. Miyata, C.A. Kamm, G.M. Kuhn, B. Yoon, R. Chellappa, S.Y. Kung. *Hierarchical Neural Networks for Learning Musical Structure*. Proc. of the 1993 IEEE-SP Workshop. IEEE, NY, NY, USA; 1993 593 pp.216-225.

D. Cope. *Computer Modeling of Musical Intelligence in EMI*. Computer Music Journal, vol. 16, no. 2; pp.69-83, Summer 1992.

Y. Freund, M. Kearns, D. Ron, R. Rubinfeld, R. Schapire, L. Sellie. *Efficient Learning of Typical Finite Automata from Random Walks*. Proc. 25th Annual ACM Symposium on the Theory of Computing, pp.315-324, 1993.

C. de la Higuera. *Characteristic Sets for Polynomial Grammatical Inference*. Machine Learning 27, pp.125-138, 1997.

M.C. Mozer. *Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-Scale Processing*. Connection-Science. vol. 6, no. 2-3; pp.247-280, 1994.

L.R. Rabiner, B.H. Wang. *An Introduction to Hidden Markov Models*. IEEE ASSP Magazine. pp.4-16, January, 1996.

D. Ron. *Automata Learning and its Applications*. Dissertation, Hebrew University, 1995.

A. Stolcke. *Bayesian Learning of Probabilistic Language Models*. Dissertation, U. California, Berkeley, 1994.

B. Thom. *Predicting Chordal Transitions in Jazz: The Good, the Bad, and the Ugly*. Proceedings of the 2nd International Joint Conference on Artificial Intelligence Music and AI Workshop, 1995.