

Learning musical pitch structures with hierarchical hidden Markov models

Michèle Weiland, Alan Smaill, Peter Nelson

▶ **To cite this version:**

Michèle Weiland, Alan Smaill, Peter Nelson. Learning musical pitch structures with hierarchical hidden Markov models. Journées d'Informatique Musicale 2005, Association Française d'Informatique Musicale; Centre de recherche en Informatique et Création Musicale, Jun 2005, Saint-Denis, France. hal-03114882

HAL Id: hal-03114882

https://univ-paris8.hal.science/hal-03114882

Submitted on 19 Jan 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

LEARNING MUSICAL PITCH STRUCTURES WITH HIERARCHICAL HIDDEN MARKOV MODELS

Michèle Weiland, Alan Smaill, Peter Nelson
University of Edinburgh, Scotland, UK
{M.Weiland, A.Smaill, P.Nelson}@ed.ac.uk

ABSTRACT

In this paper we attempt to demonstrate the strengths of Hierarchical Hidden Markov Models (HHMMs) in the representation and modelling of musical structures. We show how relatively simple HHMMs, containing a minimum of expert knowledge, use their advantage of having multiple layers to perform well on tasks where flat Hidden Markov Models (HMMs) struggle. The examples in this paper show a HHMM's performance at extracting higher-level musical properties through the construction of simple pitch sequences, correctly representing the data set on which it was trained.

1. INTRODUCTION

Modelling and representing music computationally is a task that has been the aim of many research projects in recent years; it is also the principal aim of this research to use stochastic processes to learn musical structures from existing musical data sets. Unlike expert systems, rule bases or generative theories, our approach is based on the idea that musical properties should be extracted from data sets, while as little musical expert knowledge as possible is encoded in the system's structure and parameters. We decided to use stochastic processes to model the dependencies of events in musical data using Hierarchical Hidden Markov Models. Standard Hidden Markov Models are efficient at modelling local dependencies in data sequences, but cannot represent those of a large-scale nature in an efficient way. Local dependencies are important in musical data for the encoding of style signatures [2], but a notion of large-scale dependencies is essential for the creation of believable musical structures; Hierarchical Hidden Markov Models can represent both the local and the large-scale dependencies in a data set. We would like to show that, although in our approach the models do not have a specific knowledge about the data they are processing, they are able to extract important musical notions. They might also demonstrate more creativity in the generation of new musical material than rule-oriented systems. Work has been done on metric and rhythmic as well as pitch structure; however this paper presents the results from the experiments with pitch structure only.

The remainder of the paper is structured as follows: section 2 gives background information on Hidden Markov Models and hierarchical structures in music; section 3 introduces the concept of Hierarchical Hidden Markov Models in more detail; section 4 gives an example of the use of HHMMs in modelling pitch sequences; section 5 discusses the results of the previous section; section 6 briefly outlines future steps in this research project.

2. BACKGROUND

Data sequences from any area of the real world are characterized by the local dependencies of their elements or observations. These observations can be discrete or continuous; they are elements of a knowledge space that represents the part of the world a data sequence belongs to. A successful way of modelling many data sets is by using stochastic processes, namely Hidden Markov Models (HMMs). They have been used successfully "in many applications in artificial intelligence, pattern recognition, speech recognition, and modelling of biological sequences" [1]. A HMM is a Markov chain, a general statistical tool that models the dependencies of states in a system, with two layers: the hidden layer, representing the non-observable states a system can be in, and the layer of observations that can be made by a system, depending what state it is in. Rabiner [5] gives a detailed overview of HMMs and their application in speech recognition, an area where HMMs have been rather successful.

2.1. Inference with HMMs

Three fundamental problems can be solved using a HMM: firstly, a data set can be reflected by adjusting a model's parameters, ie. the probability values for the transitions between states and the emissions of observation symbols; secondly, given a trained model, the overall probability of a sequence can be calculated; thirdly, given a trained model and an observation sequence it is possible to extract the sequence of hidden states that have most likely generated the observation sequence. The algorithms behind those problems are mathematically well-founded and offer a great versatility for the use of HMMs. In addition to the analysis abilities, a HMM can also be used as a generative model by emitting symbols while traversing the structure according to its parameters (see section 4.3).

2.2. Hierarchical Structure in Music

However, although HMMs perform well in areas such as speech recognition, language or DNA modelling, or even small-scale modelling of music, they lack performance when it comes to large-scale musical structure. Music can be seen as a composition of hierarchical structures built in time, every element in a score being dependent on its relative place in the score.

Lehrdahl and Jackendoff [4] divide the hierarchical components in music into four groups: grouping structure; metrical structure; time-span reduction; prolongational reduction. This research concentrates on aspects of the first two groups, namely the segmentation of music into phrases and sections, and the subdivision of music into weak and strong beats that give a musical work its metre.

The notion of music being hierarchical structures built in time, resulted in the idea to use Hierarchical Hidden Markov Models, instead of the standard HMMs. The additional level of expressiveness helped to overcome problems that standard models could not easily solve.

3. HIERARCHICAL HIDDEN MARKOV MODELS

Hierarchical Hidden Markov Models (HHMM) were first introduced by Fine et al. [3] as an extension of the standard flat HMM. Contrary to a HMM, a HHMM has "multiple levels of states which describe input sequences at different levels of granularity" [6] and can be thought of as having a tree structure: the nodes of the tree are the model's states, the edges are the transitions between states. HHMMs use the same methods of inference as standard HMMs, only with adjusted algorithms.

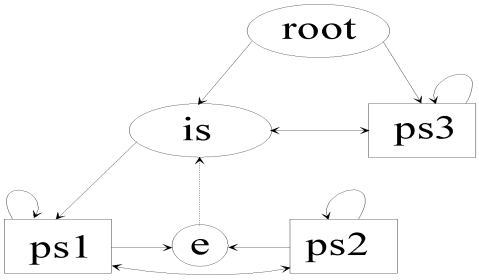


Figure 1. Example of a three-level HHMM.

3.1. States

There are three different types of states in a hierarchical HMM: internal states, production states and end states; each of them have different purposes in the model. The internal states are the tree's internal nodes, they contain information about transition probabilities to brother or child states and they are self-contained probabilistic models. Production states are the tree's leaves, they are the only states in the model that can actively emit symbols. Because production states are contained in internal states, an internal state "emits" all the symbols that its production states emit; internal states emit sequences of observations rather than single symbols. End states force, if activated, the return from one level to the next higher level. Figure 1 shows an example of a three-level hierarchical HMM with an unbalanced tree structure; the states `root` and `is` are the model's internal states, `ps1`, `ps2` and `ps3` are the production states and `e` is the end state.

3.2. Transitions

Walking through the tree structure of a HHMM is achieved by vertical, horizontal and forced transitions. The vertical transitions initially activate an internal node's child [1] ; this activation starts at the root node and is continued all the way to the lowest, the production state level of the tree. Once the production state level is reached, horizontal transitions allow movement on that level among brother states until an end state is reached and a forced transition gives control back to the parent states. From there, horizontal activations continue until a new internal state is reached and the process starts again with a vertical transition.

3.3. Level of Complexity

A major problem with HHMMs is the level of complexity caused by their actual strong point, the hierarchical structure. The algorithms introduced by Fine et al. [3] are computationally very expensive: for instance for the training of a model every single possible path has to be calculated for every possible length of the training sequence. The runtime increases cubically depending on the number of states in the model and the size of the training set. The algorithms are therefore absolutely impractical and untraceable.

3.4. Improved Performance

An alternative way of dealing with inference and training in hierarchical HMMs has been found by Daan Wierstra [7]; he devised algorithms that use the flat equivalent of a HHMM whenever possible, switching between flat and hierarchical structures. The computational advantage is enormous, improving performance to linear time depending on number of states and training set size [2] . Although the hierarchical structure is given up during some parts of

[1] The terms parent, child or brother represent the relationships of states in a hierarchical structure: brother states are states that are situated on the same level in the hierarchy; a state q's children are all the states that can be reached from state q, and are situated one level down in the hierarchal structure; a parent state is situated directly above a certain state in the hierarchy. The root state of the hierarchy doesn't have a parent; the production states, being the tree structure's leaves, don't have any children.

[2] The runtime performance is improved from $O(NT^3)$ (using the original algortihms) to $O(N^2T)$ [7], N being the number of states in the model and T being the size of the training corpus.

the algorithms, it is taken into consideration for the important processes, the parameter estimation. In order to transform a hierarchical HMM into its flat equivalent, the HHMM needs to be in minimally self-referential form, i.e. self-referential transitions are only allowed on production state level. Sections 3.5 and 3.6 are detailed expositions of the transformation process, outlined in [7].

3.5. Minimally vs. maximally self-referential HHMM

Maximally self-referential (MaxSR) HHMMs are models that have self-referential loops on internal states, i.e. internal states have horizontal transitions that point back to the same state. Contrary to MaxSR HHMMs, minimally self-referential (MinSR) HHMMS are not allowed to have self-referential loops on internal states, although they are permitted on production states.

While converting an HHMM from maximally to minimally self-referential, the equality of both models needs to be respected, i.e. sequences generated by both models need to have the same overall probability. The fundamental difference between a MinSR and a MaxSR HHMM is that if a model is minimally self-referential there is only one path through that model that connects two production states. This property allows the transformation of a MinSR hierarchical HMM into a flat HMM. The new horizontal transition probability between the states i and j in the MinSR model, both substates of q (and with i and j possibly being the same state, but j not being an end state), is:

$$a^{q*}_{(i,j)} = a^{q}_{(i,j)} + \left(a^{q}_{(i,end)} \cdot a_{(q,q)} \cdot \pi^{q}_{j}\right) \tag{1}$$

The new horizontal transition probability between i and j is the existing transition probability between those states plus the probability to go from state i to state j via their parent state: the product of the probabilities from state i to the end state, the self-referential loop in the parent state and the vertical activation of state j. The new transition probabilities towards *end* states need to balance out the fact that internal states are not allowed to have self-referential loops. The old probability of an internal state loop therefore has to propagate back through the system:

$$a^{q*}_{(i,end)} = a^{q}_{(i,end)} \cdot \left(1 - a_{(q,q)}\right) \tag{2}$$

After updating the horizontal transition probabilities between state q's children, simply set the probability of q's self-referential loop to zero:

$$a^{*}_{(q,q)} = 0 \tag{3}$$

After calculating the new probabilities, it is necessary to normalize the new values:

$$A^{q*}_{(i,j)} = \frac{a^{q*}_{(i,j)}}{\sum_{j=1}^{N} a^{q*}_{(i,j)}} \tag{4}$$

3.6. HHMM Flattening

When a HHMM is in minimally self-referential form each pair of production states in the model is connected by one single path. The flat model does not contain any internal or end states; therefore transforming the HHMM into a HMM means building a HMM from the production states only, calculating the new *direct* transitions that have not been present in the HHMM. The transitions between brother states, i.e. children of the same state i, will remain unchanged. All other transitions that were possible in the HHMM between production states, via end and internal states, need to be computed thus:

$$a^{*}_{ij} = a_{i,end} \cdot a_{q_i q_j} \cdot \pi^{q}_{j} \tag{5}$$

i.e. the probability of going from (production) state i to (production) state j is the product of the horizontal probability a of going from state i to the end state, from state i's to state j's parent, and the vertical probability π of activating state j.

The vertical transition probabilities need to be transformed into initial activation probabilities by computing the product of the vertical probabilities from the $root$ state to every production state q^D:

$$\pi^{*}_{q^D} = \pi^{q^{D-1}}_{q^D} \cdot \pi^{q^{D-2}}_{q^{D-1}} \cdot \ldots \cdot \pi^{root}_{q^2} \tag{6}$$

(with D being the production state level). Figure 2 gives an example of a minimally self-referential HHMM and its transformation into an equivalent HMM.

3.7. Technical vs. Logical Equivalence

It is important to understand that, although *technically* the hierarchical structure of the HHMM is lost during the flattening process because the tree structure is given up, *logically* the hierarchical ideas are still present in a HMM that has been built from flattening a HHMM; the models are equivalent regarding the information they can learn. However, the tree structured HHMM is restored after the training process as a way to conceptualize the hierarchical structures in the training data, because the flat model cannot represent hierarchical structures in the same logical way than a HHMM can; the models are not equivalent in the way they conceptualize hierarchial structures. Using the flat version of the model only would mean a loss of information regarding the probabilities between the different levels of the hierarchy, which provide an additional view of the internal structure of a data set.

3.8. Inference and Training

In order to fully understand the ways in which HHMMs work, the methods how to solve the three fundamental problems that were mentioned earlier need to be explained briefly.

The overall probability of an observation sequence can be calculated with the Forward-Backward (FB) algorithm.

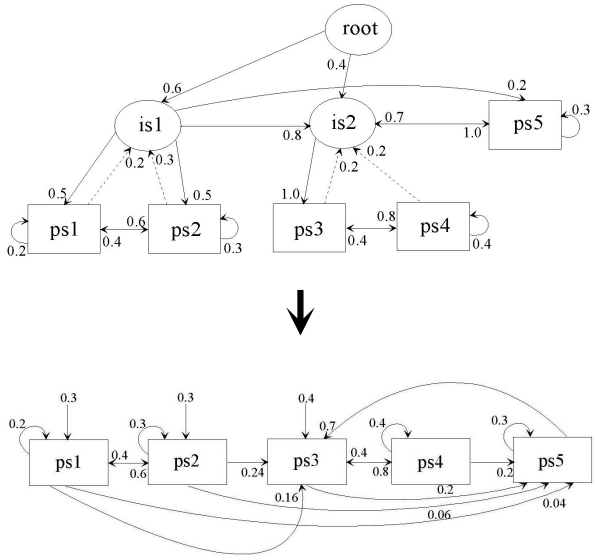


Figure 2. Example of a HHMM and its equivalent flat HMM. (The dashed transitions in the hierarchical model replace the transitions to and from the *end* states, which were omitted for clarity.) The vertical transition probabilities are converted into initial probabilities; existing direct transitions are copied into the flat model. Paths between two production states are transformed into direct transitions by adapting the probabilities according to (5).

This algorithm calculates the probabilities for each subsection of the sequence, ie. starting at the beginning of the sequence and increasing the length until the end of the sequence is reached, given the parameters of the HHMM. This way the context of each symbol is taken into consideration and the true overall likelihood of the sequence can be calculated. For the Forward-Backward algorithm, a hierarchical HMM is converted into a flat model; this conversion does not have any effect on the results, as during the conversion the hierarchical probabilities are incorporated into the flat HMM.

Finding the sequence of states that have most likely emitted a given observation sequence is achieved with the Viterbi decoding. For every symbol of the observation sequence the most likely production state, as well as the path that leads to it, is calculated, checking all the possible paths through both horizontal and vertical transitions. Again, these calculations are dependent on the context of the symbol, allowing the extraction of the state sequence that overall would be most likely.

Training a HHMM, ie. updating or adjusting the model's parameters to reflect a training set, is the most complex and important process. By scanning through the training corpus the Expectation-Maximization (EM) algorithm changes the vertical and horizontal transition probabilities between states, as well as the symbol emission probabilities in the production states in order to maximize the overall probability of the training set given the model. By calculating at each time step the probabilities of entering and leaving states, and the probabilities of moving from one state to the next, new transition probabilities can be estimated. Emission probabilities for a symbol can be estimated by extracting the likelihood of actually observing that symbol while being in a certain state.

4. MODELLING PITCH STRUCTURE

Modelling the pitch structure of a set of Bach chorales [3] is a good way to show how a structurally simple HHMM can extract important musical notions from even a small training corpus. The data that was modelled is the pitch and phrasing information from twenty-five Bach chorales - the chosen chorales were all in a major key and were transposed to Cmajor. Only the soprano from the chorales was used, and no duration information was included. The possible observation symbols for this HHMM were the opening bracket `(` and the closing bracket `)` representing the beginning and the end of a phrase respectively, the pitch symbols `C`, `C#`, `Db`, `D`, `D#`, `Eb`, `E`, `F`, `F#`, `Gb`, `G`, `G#`, `Ab`, `A`, `A#`, `Bb`, `B`, `Cb` and the symbol `#` which represents the end of a choral. No other symbols are allowed in the training set, the information that is fed into the model is limited to segments and pitch.

4.1. Choice of Model Structure

Although the inference and training algorithms run efficiently in linear time, it is still preferable to use simple models with a minimum of states, transitions and observation symbols to keep the runtime in a reasonable frame. The HHMM that models the pitch and phrase structure of Bach chorales is a three-level model; the top two levels of this model are shown in Figure 3. The motivation for having three different internal states to represent the phrases of the melodies is the importance of a distinction between the start, the body and the end of a melody; in this representation special attention is given to the start and the end of a melody, as they are both important in defining the pitch structure of a piece, but at the same time generally have very different structures. This model defines that a chorale has one start segment, one or more body segments (which can be seen as being less important in the overall structure), and one end segment. On the second level,

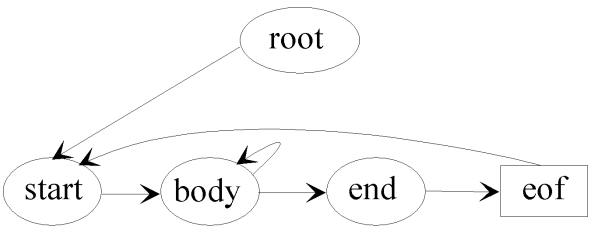


Figure 3. Top two levels of pitch/phrase structure HHMM.

there are three internal states *start*, *body* and *end*, and a

[3] The set of machine readable Bach chorales used here is available online from ftp://i11ftp.ira.uka.de/pub/neuro/dominik/midifiles/bach.zip.

production state *eof*. The internal states roughly represent the phrases in a musical piece; the piece has one start phrase, one or more body phrases and an end phrase. The *eof* production state emits the *end of choral* symbol `#`. On the third and lowest level, each internal state that represents a musical phrase contains one production state for each possible observation symbol; there are two phrasing and twelve pitch symbols (enharmonics are represented by one state) per phrasing state. Because every production state only represents one symbol, the symbols' emission probabilities are set to 1.

4.2. Parameter Initialization

Setting the initial values of all vertical and horizontal transitions is extremely important and crucial to the model's overall performance. Say for instance that all the transitions of the HHMM described in the previous paragraphs were set to random probability values at the start of the experiment; the model is completely interconnected, with no zero transitions. After updating the model's parameters with the EM procedure, it will quickly become clear that initializing all transitions randomly without providing the EM algorithm with any guidelines through the model cannot deliver satisfactory results: because the three phrasing states *start*, *body* and *end* are essentially equivalent, the EM algorithm will regard two of the three states as obsolete and set every transition towards these states to zero. Therefore, to avoid such problems, a minimum of domain knowledge will have to be included in the model structure.

With the basic layout of the HHMM, the initial vertical and horizontal transitions were set so as to force a certain path through the tree structure; simple notions such as "the music has to start with a start phrase" and "a phrase has to start with the `(` symbol" were encoded by setting certain transitions to zero, and not allowing the EM algorithm to update those values. Table 1 shows the model parameters as production rules (with transition probabilities in brackets) after training the HHMM on the set of Bach choral melodies. The table only shows the probabilities of those production states in the hierarchical model that can still be reached after training, ie. all transitions to and from production states whose observation symbols do not appear in the training data are set to 0 and are therefore not listed in the table; this explains for instance the absence of `F#` in the *end* part on the table. All the remaining pitches and their corresponding production states that are not listed in the table were also simply made redundant by being made unreachable.

4.3. Training results

Looking at the horizontal transition probabilities in the table several simple results can be extracted, such as the fact that all the chorales in the training set started on a `C` in the melody and ended their first phrase mostly on a `G`. However also musically more interesting properties are contained: according to the results, stepwise motion of

start						
(	→	C,100				
C	→	G,50	B,22	C,14	D,7	A,7
D	→	C,75	A,25			
E	→	D,75	C,25			
F	→	E,100				
F#	→	G,100				
G	→	),43	A,22	E,14	F,14	G,7
A	→	G,45	A,28	F#,9	B,9	),9
B	→	A,75	C,25			
)	→	end,100				
body						
(	→	G,46	E,22	D,18	C,10	
		A,4				
C	→	B,35	),30	D,23	A,12	
D	→	C,32	E,32	),16	D,13	G,3
		F#,3				
E	→	D,40	E,20	),18	A,7	F,7
		G,5	C,3			
F	→	E,78	F,11	G,11		
F#	→	G,100				
G	→	E,30	A,27	),23	F,20	
A	→	B,39	G,39	A,14	C,4	E,4
B	→	C,61	A,34	),5		
)	→	(,75	end,25			
end						
(	→	C,57	G,29	A,14		
C	→	),41	D,24	B,23	C,12	
D	→	C,73	E,27			
E	→	D,85	F,15			
F	→	E,80	D,20			
G	→	A,51	F,49			
A	→	G,57	B,29	F,14		
B	→	C,51	A,49			
)	→	end,100				

Table 1. Horizontal transition probabilities after training. The pitches and phrase symbols represent the production states; *end* represents the end state of the level that forces the return back to the next higher level. The symbols on the RHS of the rules are the events that can be reached from the symbol on the LHS of the rule. The numbers represent the transition probabilities from the LHS to the RHS in percent.

the soprano is clearly preferable, leaps seem to be rare. If there is a leap in the melody, it is mostly followed by stepwise motion in the opposite direction. This is in agreement with the well-known rules of melodic movement in chorales.

In order to get a better overview of the trained HHMM's performance, a set of one hundred new chorales was generated. The generation process is simple: walk through the tree structure, at every crossroad randomly decide which path to take or which symbol to emit, and continue doing so until the *end of choral* symbol # is emitted. This process introduces a certain degree of "creativity"; as every decision is taken randomly the newly composed sequences are, almost certainly, different from the sequences in the training set, but still related to what the model has learned.

A few examples of soprano sequences generated by the HHMM described earlier are shown in Figure 4.

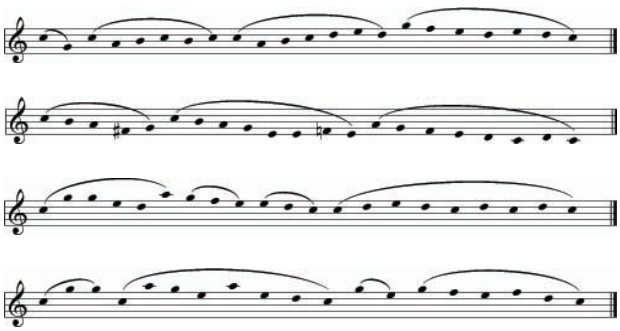


Figure 4. Four examples of soprano lines generated by the HHMM.

These sequences are a sample set, chosen because these short examples reflect the musical notions the HHMM has incorporated after training. The length of these examples is not typical; because the sequences are generated randomly, there is no actual way of controlling the lengths, although the horizontal transition probabilities between the states `)`, `(` and `end` certainly give preference to start a new body phrase after finishing one (see Table 1). The stepwise motion of the melody is clearly preferred in the pitch structure, as discussed earlier. But more promisingly, all the sequences that were generated end their phrases in cadences, and each melody concludes in a perfect cadence.

Using the same model structure, the experiment was also run on the bass line of the same Bach chorales. As expected, the HHMM learns that leaps of fourths and fifths are more common in a bass line, while stepwise motion is still important. The ability to create cadences at the right moments is still given. An example of a bass line is shown in Figure 5.



Figure 5. Example of a bass line generated by the HHMM.

5. DISCUSSION AND CONCLUSION

This paper aims to discuss the suitability of HHMMs for representation and modelling of music, and its possible advantages in performance over standard HMMs. Evaluating the HHMM's performance based on the results presented, the approach most certainly demonstrates the ability to model and generate music in a 'musical' way with very little given information about the data set. The hierarchical structure of the HHMMs plays a major part in the success of this approach. In a flat HMM, the hidden states can be interconnected in any possible way, but there is only one hidden layer. In the hierarchical HMM however, every level is connected among itself *and* the levels above and below it.

Going back to the example of pitch structure modelling from the previous section: in a flat HMM[4], a natural choice for the model structure would be one set of twelve pitch states and two phrasing states; the model would have no way to distinguish between different phrases, and between different phrases at different times in a piece. If it cannot distinguish between those phrases, it cannot learn that phrases in different parts of a piece of music have different properties, ie. although the first phrase of a Bach choral ends on the dominant most of the time, this is not true for the last, ending phrase in a piece. Training such a HMM on the same data as the HHMM, it would not be able to represent the overall musical structure as accurately as the HHMM.

The extra level of expressiveness that a HHMM offers can be shown with that same example: the relationships between the phrases of a piece are modelled on one level; the level below models the dependencies of the events inside those phrases. When generating new sequences with the HHMM it is almost as if two processes are running at the same time: on one level, the phrases are constructed, on another level their events are created - and both process layers are contained in the root state. The generation process thus becomes more natural as it is happening simultaneously on different levels, creating a sequence as a whole, rather than as a chain-like process that constructs a sequence one symbol at a time.

However, although HHMMs are very powerful for representing hierarchical structures, they are only powerful if their potential is used in the right way, ie. if appropriate model structures are chosen to represent certain aspects of musical knowledge. A big challenge therefore is to find good representations of these aspects that allow models to be kept small and manageable; the bigger the models, the harder it is to find the best possible structures.

As a conclusion it can be said that, although hierarchical HMMs are complex in their implementation, and can quickly become unmanageable and inefficient if their structure is not chosen ideally, they certainly are a step

[4] By flat HMM we mean a HMM that does not incorporate any idea of conceptualizing data hierarchically; we do not refer to the HMM that can be constructed from flattening a HHMM, which would learn the same information as the HHMM, only lacking in accurate hierarchical representation.

forward in the attempt to generate believable large-scale musical structures.

6. FUTURE WORK

Future work on this research project will concentrate on further modelling of music's structural properties, for instance harmonic structure, or the combination of metrical and pitch structure. As current results demonstrate the suitability of HHMMs for learning large-scale musical structures, and properties such as cadences, future efforts will go into trying to find further simple representations of musical structures that are able to learn higher-level musical properties. It is hoped that a network of HHMMs, each representing a different musical characteristic, can be built, allowing the generation of more complete musical examples that can show the potential of hierarchical HMMs in learning music and being 'creative'. Additionally, we will aim at expanding the project towards using Input-Output HHMMs, which model the dependency of an output sequence not only on a series of hidden states but also on a given input sequence, thus giving an advantage of modelling the relationships between different structural properties.

7. REFERENCES

[1] Yoshua Bengio. *Markovian Models for Sequential Data*. Neural Computing Surveys 2. 1999.

[2] David Cope. *Computers and Musical Style*. Oxford University Press. 1991.

[3] Shai Fine, Yoram Singer and Naftali Tishby. *The Hierarchical Hidden Markov Model: Analysis and Applications*. Machine Learning, Vol.32, No.1. 1998.

[4] Fred Lerdahl and Ray Jackendoff. *A Generative Theory of Tonal Music*. MIT Press, Cambridge. 1983.

[5] Lawrence R. Rabiner. *A Tutorial On Hidden Markov Models and Selected Applications In Speech Recognition*. Proceedings of the IEEE, Vol.77, No.2. 1989.

[6] Marios Skounakis, Mark Craven and Soumya Ray. *Hierarchical Hidden Markov Models for Information Extraction*. Proceedings of the 18th International Joint Conference on Artificial Intelligence, Acapulco, Mexico. 2003.

[7] Daan Wierstra. *A New Implementation of Hierarchical Hidden Markov Models*. Master's Thesis, Utrecht University. 2004.