Chorale Harmonization with Weighted Finite-state Transducers

Jan Buys and Brink van der Merwe
Computer Science Division, Department of Mathematical Sciences
Stellenbosch University, South Africa
janbuys@ml.sun.ac.za, abvdm@cs.sun.ac.za

***Abstract*—We approach the task of harmonizing chorales through style imitation by probabilistically modelling the harmony of music pieces in the framework of weighted finite-state transducers (WFSTs), which have been used successfully for probabilistic models in speech and language processing. The framework makes it possible to place domain-specific regular constraints on generated sequences, and to integrate models of different levels of complexity. We divide the harmonization generation process into different steps, each performed by inference through transducers. We present a method for four-part harmonization that models vertical and horizontal structure in the generated harmonizations. The weights in our transducers are learned by maximum likelihood estimation from a corpus of chorales. The predictive power of own model, as measured through entropy, is competitive with that of existing approaches.**

I. Introduction

Music is usually composed through a creative process. However, all pieces of music have structure, and a composer is usually constrained by the style of his composition. Established rules and principles that give the music certain aesthetically pleasing qualities should be followed. However, the characteristics of a good music piece cannot be fully described by such rules. The main reason for this is that a music piece should exhibit acceptable structure at a local and global level. There should be a fine balance between various musical qualities in the piece.

The harmony of a music piece, i.e., the structure of notes played simultaneously, is central to this musical structure. The harmony, usually described by chords, can be seen either as an observed variable of a piece with multiple voices, or as a latent variable of a melody. In music pieces with multiple voices, both the chords formed by the voices at each time-step (also called the vertical structure, due to the way music is written) and the melodic structure of each of the accompaniment voices (called the horizontal structure) are important. In this paper we will be concerned with the particular style of harmonization in 17th-century 4-part chorales, exemplified by the chorales of JS Bach. Chorale harmonization is an important task in Western classical music, and is studied by all students of music theory. The principles employed in chorale harmonization are transfered to many other composition tasks which involve harmony.

In this paper we present a probabilistic model to generate harmonizations for given melodies, using machine learning techniques. We focus on style imitation of chorales by JS Bach, though our model can also be applied to other genres. Bach chorales have been used extensively in music modelling (see for example [1], [2], [3]), due to their abundance, simplicity and good melodic and harmonic form. Our model is predictive, assigning a non-zero probability to every possible sequence over the given alphabet.

The harmonization procedure that we propose models the different steps in the harmonization process, similar to those followed by human composers, in a full probabilistic setting. Each step is performed by inference through a weighted finite-state transducer (WFST) cascade. Separate models are trained for major and minor pieces, as there are significant differences between their musical characteristics. Our model generates a good approximation of the real harmonization, modelling both vertical and horizontal constraints.

In the next section we give some musical background and discuss related work. Section 3 defines weighted finite-state transducers and related algorithms required for our approach. Section 4 presents the harmonization model we propose, while section 5 discusses the implementation of our model. We discuss the evaluation of our system in section 6, and give conclusions in section 7.

II. Background and Related Work

A. Musical Notation

The fundamental units of a music piece are *notes*. A note is a single sound, represented by *pitch* — how high or low the sound is, and *duration* — how long the sound is held. In a standard classical music piece, the pitch and duration of notes are governed as follows:

Pitches are named by their *pitch classes*. There are 12 classes, namely C, C#, D, D#, E, F, F#, G, G#, A, A# and B, each of which forms an equivalence class. An interval of size 12 is referred to as an *octave*. A *scale* is a sequence of pitch classes defined by the starting pitch class of the scale and the intervals between pitches in the scale. The most common scale types are the major and (natural) minor scales. The *key signature* of a piece indicates the scale that forms the basis of notes of the piece. However, a piece can also have *accidentals*, notes that are not in the scale of the key signature.

The *beats* of a music piece are constant time intervals that primarily govern the start of notes in the music. The *tempo* indicates the length of those beats. The *time signature* indicates the meter of the music, the basic grouping beats into *bars*. The

rhythm of a sequence of notes describes the duration of each note in the context of the time signature of the piece. Note durations are expressed as fractions of a "whole note" of 4 beats. Typical durations are a half note, quarter note, eight note and sixteenth note.

In general, a music piece consists of a number of voices, each voice being a single time-dependant sequence of notes. The *melody* of the music piece is the most significant voice, usually the highest. The *harmony* of the music refers to the way that different notes sound simultaneously. The harmony can be described by *chords*, predefined combinations of notes in the scale of the music piece that sound well together. A chord usually consist of three pitch classes, though there are also chords with more pitch classes. The most common chord types, those that we will focus on, are the major and minor chords.

B. Algorithmic Composition

Algorithmic composition, i.e., composition by formalizable methods, has a long tradition, and numerous procedures have been investigated [4]. The most common limitation that these approaches have is the inability to generate longer pieces of music that exhibits acceptable overall structure. One broad approach to algorithmic composition is the application of rules or algorithms chosen by the composer or programmer to create new pieces of art [5]. However, composers seldom publish the formalizable ideas that they use in their compositions [4]. The other approach is to construct a generative theory to describe music pieces in a given style [1]. This generative theory can then be used to generate new pieces of music in the given style. Just as in language modelling, one can distinguish between two approaches to such generative theories. In the first approach, knowledge engineering, rules and constraints are explicitly encoded in some logic or grammar. In the second approach, empirical induction, parameters of a statistical model are learnt from existing compositions.

There are clear parallels between the development of generative models for language and for music. However, in computational linguistics the focus is primarily on using models for analysis, while in music modelling research focuses on generation. Conklin [6] argues that the problem of music generation can be made equivalent to that of sampling from a statistical model; Models that are able to explain the structure of music pieces will also be able to generate acceptable original music pieces. A reason why this might be the case (it is not so in language modelling) is that in music the semantics (meaning) lies primarily in the structure of the music, while in language words can have meaning that lies outside the structure, by referring to objects or actions. However, more research into this relationship, and its implication in evaluating music generation systems, is needed.

Markov models have been used widely in musical style imitation, since they are simple and efficient in training and inference. To model melodies, higher order and variable order Markov chains are usually used (see, for example [7], [1], [3], [8]). The best results found are found by using a middle ground between Markov chains of low order that do not constrain the structure of generated music sufficiently, and Markov chains of high order that reproduce large fragments of the music pieces used for training.

C. Harmonization Models

Allan and Williams [2] applies hidden Markov models (HMMs) to harmonize chorale melodies. An HMM takes the melody notes as the observed sequence and the possible harmonizations (chord configurations) as hidden states. The best harmonization for a given melody is obtained from the HMM. A second HMM is used to model ornamentation, i.e., to add notes with a duration other than that of the beat to the generated harmonization voices. Ornamentation smooths the movement between notes and adds some variation. The MySong automatic accompaniment system [9] uses a similar HMM approach to generate chords to accompany a melody sung by the user.

A probabilistic graphical model approach to harmonization is proposed in [10]. Domain knowledge can be included in the model, and different levels of hidden variables are used to model non-local dependencies in the chord progressions. Specifically, it is found that a tree-structured graphical model for modelling the roots of chords over a given melody has more predictive power than an HMM model for the same task. However, a similar advantage was not found when modelling other voices in the harmony given the chord roots.

III. Weighted Transducers

Weighted finite-state transducers are automata that can be be used for the probabilistic modelling of discrete sequences. Important motivations for the use of WFSTs include the uniform representation of models and the existence of efficient inference algorithms that can be applied to them [11]. WFSTs have been used successfully in speech and language processing (see for example [12]). We now define transducers and the operations that we need to use them as probabilistic models. Then we show how Markov chains and hidden Markov models can be represented as WFSTs.

A. Finite-state Transducers

A *weighted finite-state acceptor* (WFSA) is a finite-state machine that accept a set of strings in the class of regular languages, assigning weights to the accepted strings. It has states and edges between states. Each edge is labeled with a symbol in the alphabet of the language, and a weight. A WFSA accepts a string if there is a path from a start state to a final state such that the concatenation of the symbols on the edges along the path yields the string. The symbol ϵ, denoting the empty string, can also be used as an edge label. The weight assigned to a path is the product of the weights on the edges along that path. The weight of a string is the sum of the weights of all the paths that yields that string.

A *weighted finite-state transducer* is a finite-state automaton similar to a WFSA, where each edge has an input and an output symbol. A WFST assigns weights to accepted pairs of

input-output strings, and can also be considered as a device which transforms a string in one regular language to a string in another regular language. A string is a sequence of alphabet symbols. Below we will refer to strings as sequences.

Formally (see [11]), a weighted finite-state transducer T over a semiring $\mathcal{K}$ is a tuple $(\Sigma, \Omega, Q, E, I, F, \lambda, \rho)$ given by: An input alphabet Σ; an output alphabet Ω; a finite set of states Q; a finite set of weighted transitions E contained in $Q \times (\Sigma \cup \epsilon) \times (\Omega \cup \epsilon) \times \mathcal{K} \times Q$; a set of initial states $I \subseteq Q$; a set of final states $F \subseteq Q$; an initial weight function λ; and a final weight function ρ.

Semiring abstraction allows us to define automata representations and algorithms over different weight sets and algebraic operations. A *semiring* K consists of a set $\mathcal{K}$ with an associative and commutative operation $\oplus$ and an associative operation $\otimes$, with identities $\overline{0}$ and $\overline{1}$, respectively, such that $\otimes$ distributes over $\oplus$, and $\overline{0} \otimes a = a \otimes \overline{0} = \overline{0}$. Here we use the probability semiring, where weights are probabilities and the usual summation and multiplication operators are used. When we construct a transducer the weights do not have to be normalized; we just need to normalize them when we compute probabilities or perform transducer operations dependent on the value of the weights.

Transducers that represent different levels of representation in a model are combined with the operation of *composition*. The composite transducer $T = T_1 \circ T_2$ accepts the sequence pair $(A : C)$ if and only if there exists a sequence B such that T_1 accepts the pair $(A : B)$ and T_2 accepts the pair $(B : C)$. The weight assigned to $(A : C)$ is the sum, over all possible values of sequence B, of the product of the weights given by T_1 and T_2. Composition can also be extended to a cascade of more than two transducers. *Right projection* is a unary operation on a WFST that yields a WFSA that accepts exactly the output sequences that the WFST can produce. The weight of a sequence B in the acceptor is the sum of the weights of all pairs $(A : B)$ in the WFST. Similarly, the *left projection* gives a WFSA over the input sequences of the transducer.

We will use WFSTs by the process of *application*, obtaining the result of the transformation of some input by a transducer or a cascade of transducers. The application can be *forward*, when an input sequence to the transducer is given and we want to find an output sequence, or *backward*, when the output sequence of the transducer in the cascade is given, and we want to find an input sequence that can be transformed to that output sequence. To apply a sequence to a transducer, the sequence is converted to an identity WFST that accepts only that sequence, with weight 1. That WFST is then composed with the given transducer, and the applicable projection of the composite transducer is obtained (right projection for forward application, and left projection for backward application). We can then sample from, or find the most likely sequence of, the resulting WFSA.

B. Markov Models

A *Markov chain* (MC) is a stochastic chain over a discrete number of states. The probability of a state in an nth-order Markov chain is dependent on the values of the previous n states in the chain. A sequence of symbols generated by an MC represents the states of the chain. Therefore, in the sequence $q_1, q_2, ..., q_m$ the following assumption holds if $t \geq n$:

$$P(q_{t+1}|q_t, ..., q_1) = P(q_{t+1}|q_t, q_{t-1}, ..., q_{t-n+1})$$

We can represent a Markov chain as a WFSA as follows: The alphabet of the WFSA is the set of symbols representing the state space of the MC. Each state of the WFSA encode the history of the previous n states in the MC. A transition in the WFSA is labeled with a symbol representing the next state of an MC transition, and its weight represents the probability of that transition. It follows from this representation that, when we ignore probabilities, MCs generate a class of languages (with the state names as alphabet symbols) that is a strict subclass of the regular languages.

We can learn the weights of this WFSA by maximum likelihood estimation, as for an MC. The $(n+1)$-gram counts of the sequences in the training data being modeled are the sufficient statistics. The probability of a transition between states q_t and q_{t+1}, given the state history, is:

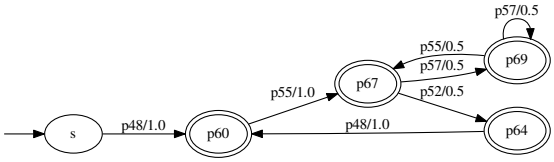
$$\frac{\text{count}(q_{t-n+1}, ..., q_t, q_{t+1})}{\text{count}(q_{t-n+1}, ..., q_t)}$$

To make the model predictive, we add smoothing to the Markov chains we use in our models. We use Katz's back-off model [13], a smoothing method often used in language models for speech recognition. In a higher-order MC, when an n-gram does not occur, we recursively back off to the highest-order MC for which the corresponding m-gram suffix of the n-gram has a non-zero probability. This is an appropriate smoothing technique for music sequences, due to its similarity to the variable-order Markov chains that have been used successfully for melodic modelling. For the models we use here, it was sufficient to use a second-order Markov chain (a trigram model). Counts of n-grams whose frequency is lower than a threshold k (we use $k = 5$, the most common choice for k) are lowered using Good-Turing re-estimation, and the freed up fractional frequency counts are redistributed to assign back-off probabilities to lower-order MCs. Let n_r be the number of n-grams that occur exactly r times in the training data. Then the discount coefficient for n-grams, where $1 \leq r \leq k$, is

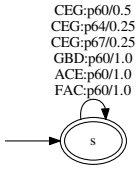
$$d_r = \frac{\frac{(r+1)\cdot n_{r+1}}{r \cdot n_r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}}$$

.

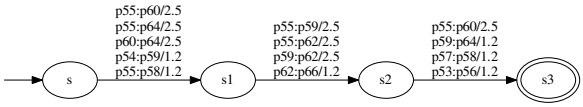
A *hidden Markov model* (HMM) [14] models the relationship between two sequences, a hidden sequence and an observed sequence. A discrete HMM can be represented by a cascade of two WFSTs. The first transducer is an MC for the hidden symbol sequence. The second transducer models the state emission probability distributions. This transducer has a single state, takes as input the hidden sequence, and gives as output the observed sequence. Every transition in this transducer has an input symbol from the hidden sequence alphabet, an output symbol from the observed sequence alphabet, and a



(a) Bass voice acceptor



(b) Chord to pitch transducer



(c) Inner voices transducer

Fig. 1. Example transducers for harmonization generation

weight representing the probability of the observed symbol given the hidden symbol. For a given observed sequence, we can find a corresponding hidden sequence by backward application to the transducer cascade. In our models, we estimate the weights in the HMM by maximum likelihood estimation (separately for the two transducers) on sequence pairs in the training data, as both the observed and the hidden sequences are known during training.

The framework of WFSTs makes it possible to place regular constraints on sequences generated by Markov chains or Hidden Markov models. For example, we can constrain the length of a sequence by constructing an acceptor of all sequences of a specific length. We then compose that acceptor with the Markov chain WFSA to get a state-machine that represents the same MC, but only accepts sequences of the required length. It is also possible to compose a Markov model with a non-Markov model that represent some domain knowledge, and which is representable as a WFST.

IV. Harmonization

We present a model to harmonize given melodies in the style of four-part chorale harmonization. Our harmonization procedure has two steps. Firstly, we find the optimal chord sequence for a given melody. Secondly, we generate three additional voices (the bass voice and two inner voices) so that the implied harmony corresponds to the generated chord sequence. In the approach proposed in [2], the chord representation includes the configuration of notes in the harmonization voices. The advantage of our model is that we are able to model explicitly the voice movement of the harmonization voices, which is important for good harmonizations.

In our harmonization generation system we generate music by performing inference through application to transducer cascades. At each step, the Viterbi algorithm is used to find the optimal sequence. Examples of the WFSTs used in the cascades are given in Figure 1. We give a schematic representation of the transducer cascades used, in Figure 2. Each transducer's input and output sequences are given. For each of the transducer cascades we also give corresponding probabilistic graphical model representations in Figure 3.

A. Chord Analysis

To model the harmony, we first analyze the chords in the music piece (see [15] for an overview of procedures). We use a template-based method to assign a chord to each beat of the music piece. The template chords we use are the 12 major chords, the 12 minor chords, and the empty chord (corresponding to no chord classification made).

For chord classification, the notes in a beat are represented by a vector. Each element in the vector represents the duration of notes in the beat corresponding to one of the 12 pitch classes. We represent our template chords similarly: The three pitch classes are each represented by the duration of a beat, but the tonic of the chord is represented by twice that value, due to its importance. We classify the notes in each beat to the template chord for which the Euclidean distance between the vector representation of the beat notes and the template chord is a minimum.

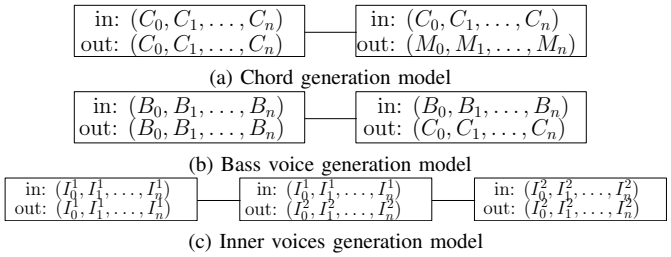
B. Chord Generation

We use an HMM approach to find the optimal chord sequence for a given melody. The chord sequence is seen as the hidden sequence, modelled with a (higher order) Markov chain, and the melody is seen as the observed sequence. The relationship between the chord and melody notes in each beat in the music is modeled.

The melody sequence symbols each represent the pitches and rhythm of a beat in the music. The chord sequence symbols are template chord names. We model chord generation with a WFST cascade, given in Figure 2a, where the first transducer is a Markov chain for chords and the second transducer is a single-state chord to melody transducer. The corresponding graphical model is given in Figure 3a. To do inference, we apply a melody symbol sequence to the cascade, and find the optimal chord sequence with the Viterbi algorithm.

C. Bass Voice Generation

To generate harmonization voices, we first generate a bass voice, and then two inner voices. In the training data, we identify the base voice as the voice that is, on average, the lowest in a music piece. In the model for the bass voice we work with a representative pitch sequence for the bass note sequence. When there is more than one pitch in the same beat, the longest or first pitch is chosen. We model the relationship



(a) Chord generation model

(b) Bass voice generation model

(c) Inner voices generation model

Fig. 2. Transducer cascades for the harmonization model

between the bass voice and the chord sequence with a hidden Markov model, in a similar way to the chord sequence and melody. Now the chord sequence is the observed sequence and the bass note sequence is the hidden sequence. Given a chord sequence, we sample from the distribution of bass notes for that chord sequence. The transducer cascade is given in Figure 2b and the graphical model in Figure 3b.

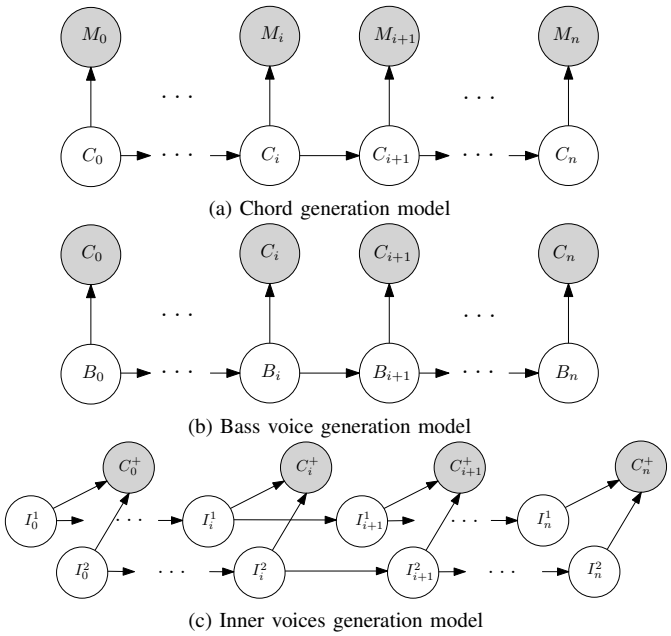
D. Inner Voices Generation

We generate two inner voices that, together with the melody and bass voices, give the implied harmony of the chord at each beat. To do this, we model the vertical constraints on, and the horizontal probability distribution over the generated voices. We use a smoothed Markov chain trained over all the inner voices in the training data to model the horizontal structure. There is usually an overlap in the ranges of the inner voices, and in training a single model for the inner voices we do not need to restrict the training data to exactly four voices.

Vertically, we model the pair of inner voice sequences so that the four voices of the harmony will together represent the chosen chords at each time-step. We restrict the range of the voices such that none of the four voices may cross each other (a lower voice may never have a higher pitch than a higher voice in the same beat). The motivation for this constraint is to remove spurious ambiguity from the model. We want to give a strong preference to inner voice pairs at a beat such that all three pitch classes of the chord should be contained in the four notes at the beat. If there is no configuration satisfying that preference, we give preference to assigning one note, in a pitch class not yet represented by the melody or bass voice of the beat, to both voices.

We want to model the two inner voice sequences given the melody, chord and bass sequences. We represent the already-known sequences with the sequence C^+, where C_i^+ encodes the melody note, bass note and chord at time-step i. The graphical model representation of the joint distribution over these variables is given in Figure 3c. The distribution factorizes as follows:

$$\begin{aligned}P(I^1, I^2, C^+) =& P(I_0^1, ..., I_n^1, I_0^2, ..., I_n^2, C_0^+, ..., C_n^+)\\ =& P(I_0^1)P(I_0^2)P(C_0^+|I_0^1, I_0^2)\\ &\bullet \prod_{i=1}^{n} P(I_i^1|I_{i-1}^1)P(I_i^2|I_{i-1}^2)P(C_i^+|I_{i-1}^1, I_{i-1}^2)\end{aligned}$$



(a) Chord generation model

(b) Bass voice generation model

(c) Inner voices generation model

Fig. 3. Graphical models for the harmonization model

As we are working with the case where $C_1^+, ...C_n^+$ is given, the conditional distribution is:

$$\begin{aligned}P(I^1, I^2|C^+) =& P(I_0^1, ..., I_n^1, I_0^2, ..., I_n^2|C_0^+, ..., C_n^+)\\ =& P(I_0^1)P(I_0^2)\frac{P(C_0^+|I_0^1, I_0^2)}{P(C_0^+)}\\ &\bullet \prod_{i=1}^{n} P(I_i^1|I_{i-1}^1)P(I_i^2|I_{i-1}^2)\frac{P(C_i^+|I_{i-1}^1, I_{i-1}^2)}{P(C_i^+)}\end{aligned}$$

We model this distribution with a transducer cascade as follows: The first transducer is a Markov chain for the first inner voice, the second transducer models the vertical structure — the acceptability of inner voices at every time step (taking the first inner voice as input and giving the second as output) and the third transducer is a Markov chain for the second inner voice. This transducer cascade is given in Figure 2c. From the probability distribution factorization it follows that the weight of a transition between time-steps $i-1$ and i in the second transducer should be $\frac{P(C_i^+|I_{i-1}^1, I_{i-1}^2)}{P(C_i^+)}$.

Let α and β be weights that indicate our preference for pure chords (all three pitch classes are represented) and impure chords (any other note combination) respectively. Here we choose $\alpha = 0.8$, and let $\beta = 1 - \alpha$. The reason for this choice is that we want to give a strong preference to pure chord representations.

Suppose $P(C_i^+) = \frac{1}{m}$, where m is the number of possible chord combinations. Let p_i be the proportion of possible inner voice combinations that represent pure chords, given the bass and melody notes at time step i. The transition weight is then $\frac{\alpha}{\alpha \cdot p_i + \beta \cdot (1-p_i)}$ for pure inner voice combinations and $\frac{\beta}{\alpha \cdot p_i + \beta \cdot (1-p_i)}$ for impure combinations.

E. Ornamentation

In general, harmonization voices are not played in blocks, at every beat in the music. Repeated notes may be combined into one longer note, and extra notes can be inserted to improve voice movement (the most common example is to insert a middle note if there is an interval of a third between two notes). This process is known as ornamentation. An HMM model for ornamentation is proposed in [2]. We implement a similar model. We first encode the pitches and rhythm of a note sequence at each beat as a single symbol. The ornamented note sequence is then modelled as the hidden sequence of the HMM, and the representative note sequence as the observed sequence. The ornamentation transducer cascade can be applied to ornament the bass voice and the two inner voices.

However, this ornamentation procedure is limited in its ability to reproduce ornamentations of quality comparable to the ornamentations in our training data. Another limitation is the inability to model parallel or diverging movement in pairs of voices, as the ornamentation of different voices is modeled independently. Excessive or uncoordinated ornamentation may decrease the quality of harmonizations. We therefore propose that further work should be done on ornamentation, building on the ability of our approach to model vertical and horizontal structure in harmonizations. We do not include ornamentation in our evaluation below.

V. Implementation

In this section we give a brief overview of the implementation of our chorale harmonization generation system. The main steps in the system are:

1) Analyse a corpus of given music pieces.
2) Learn the parameters of a WFST-based model for harmonization.
3) Generate new harmonizations for given melodies, using the trained model.

In our implementation we use the finite-state transducer package Carmel [16] for performing operations on the transducer models. Carmel can train and compose transducers, sample sequences or get sequence probabilities from transducers.

To represent music pieces our system uses MIDI, a standard music file format that represents a music piece by event messages about the music, rather that with an audio signal. In our implementation we use the Java package JMusic [17] to extract a symbolic representation corresponding to standard music notation from a MIDI file. We extract a pitch sequence and a rhythm sequence for each of the voices in a music piece. A pitch value is represented as a MIDI pitch value, an integer between 0 and 127 that represents the number of semi-tones the note is higher than the note 5 octaves below middle C. For our model, we transpose the pitches of all the training music pieces to the key of C major or A minor, for pieces in a major or minor key respectively. The rhythm sequence represents the durations of notes and rests, as well as bar separators.

We represent rhythm (note duration) values with integers directly proportional to the note duration, with 96 representing a whole note. However, as these note values are not always represented precisely in the MIDI files, we had to approximate imprecise values to the nearest discrete value in our representation. We also inferred the bar structure in the music pieces, and used that to ensure the correct alignment of notes to beats.

In our implementation we store the transducers and note sequences as text files in the format required by Carmel. We then use Carmel to perform inference by application to transducer cascades, using the models described above. Finally, the generated sequences for the harmony (in text format) is converted back to a MIDI file of the generated harmonization.

TABLE I
Average entropy of harmonization models on training and testing sets

Model	Major Train	Major Test	Minor Train	Minor Test
$P(C)$	2.503	2.796	2.748	3.171
$P(C\|M)$	1.517	1.798	1.851	2.219
$P(B)$	4.259	4.144	4.382	4.407
$P(B\|C)$	2.463	2.377	2.463	2.377
$P(I^1, I^2)$	3.727	3.957	3.859	4.182
$P(I^1, I^2\|M, C, B)$	2.886	3.137	3.090	3.330
$P(H\|M)$	6.866	7.312	7.364	7.949
$P(H\|M)$ in [2]	3.693	7.069	3.838	7.242

VI. Evaluation

The evaluation of our models is based on a publicly available corpus, in MIDI format, of chorales by JS Bach[1]. From this corpus we used 350 chorales in four-part harmony, evenly split between chorales in major and minor keys. We trained separate models for chorales in major and minor keys. For both models, the chorales were divided into a training set (60%) and a testing set (40%). Our evaluation is based on an estimation of entropy – the negative log likelihood per symbol that a model gives to music pieces in the testing set. For a sequence $S = s_1 s_2 \dots s_N$, the value $-\frac{1}{N}\sum_{i=1}^{N} log_2 P(s_i|s_1, \dots s_{i-1})$ is used.

The average of this measurement is taken over all examples in the testing set. This evaluation method has been used to evaluate harmonization in [2]. It evaluates the predictive power of a model by measuring the likelihood that is assigned by our model to compositions that we are trying to imitate. The lower the entropy, the higher the probability that our model gives to the music pieces. We compute the entropy for different components of our model separately, and then add them to find the entropy of the model. Table I gives the average entropy of our models. We include the entropy of the training and testing sets. For each of the models, we compare the result against a baseline Markov model that is not conditioned on other sequences.

The predicted sequences are represented by the following symbols: The melody, M, the chord sequence, C, and the harmonization voices, H (representing the bass voice B and inner voices I_1 and I_2). We include the results of Allan and

[1] http://www.jsbchorales.net/down/sets/jsb403.zip

Williams [2] for comparison. Note that we convert their scores from $\log$ base e to $\log$ base 2 to represent entropy.

In the evaluation of our models, we find that the entropy of the training data is in each case smaller than that of the testing data, as should be expected, but only by a small margin. This shows that our smoothed models (using Katz's back-off model) are very robust in dealing with sparse data, and performs almost equality well on seen and unseen data. In contrast, the results of [2] show a large difference between the testing and training training data. This gives some evidence that their model overfits the training data.

The results show that for each of the models an improvement is obtained over the baseline model, where the sequence is independent of other sequences. This show that our model is indeed modelling the dependencies between sequences.

The results we obtain from our model is competitive with the results of [2]. One reason our model does not perform better is that, with enough data, that model will also model movement in individual harmonization voices, as all the notes at a beat are encoded in a single symbol. However, our model should be more scalable, as we explicitly model horizontal movement in harmonization voices.

The evaluation approach we follow here allows us to evaluate a model hypotheses quantitatively and to compare different hypothesis. A limitation of this approach is that the model has to be predictive. In practice, one might generate better quality music by placing more hard constraints on the generated harmonizations. Specifically, it might be beneficial to constrain the inner voices to let only pure chord combinations be generated.

An alternative way to evaluate our harmonization system would be to let a music expert panel judge the quality of the harmonizations and their conformity to standard harmonization rules.

VII. Conclusion

In this paper, we proposed a model for the harmony of music pieces, specifically chorales, using the framework of weighted finite-state transducers. This framework is flexible and extendible, making it possible to construct models that encode different sets of dependencies and restrictions. We showed how WFSTs can model different steps in the harmonization process in a probabilistic setting, while encoding algorithmic processes that composers may be following when they compose pieces of music. The results show that our procedure is successful in modelling dependencies in the horizontal and vertical structure of the music.

For future work, models for non-local structure in the chord sequence generation model should be investigated. Specifically, tree-based approaches should be considered – a non-probabilistic tree-based approach for music modelling has been proposed in [18]. The restriction on chords types in our model can also be relaxed.

Another avenue for further investigation is the effect of removing some of the independence assumptions we made by dividing the harmonization process into different steps. Better models for non-local structure in the harmonization should also be developed. The ornamentation procedure we mentioned can be refined. Related to that, approaches to modelling parallel and diverging movement between voices in the harmonization should be investigated. The goal should be to fully model the richly structured harmonizations of JS Bach.

Acknowledgment

The first author would like to thank the financial support of the Wilhelm Frank Bursary fund and the MIH Media Lab.

References

[1] D. Conklin and I. H. Witten, "Multiple viewpoint systems for music prediction," *Journal of New Music Research*, vol. 24, no. 1, pp. 51–73, 1995.

[2] M. Allan and C. K. I. Williams, "Harmonizing chorales by probabilistic inference," in *Advances in Neural Information Processing Systems*, vol. 17. MIT Press, 2005, pp. 25–32.

[3] J. L. Trivino-Rodriguez and R. Morales-Bueno, "Using multiattribute prediction suffix graphs to predict and generate music," *Computer Music Journal*, vol. 25, no. 3, pp. 62–79, 2001.

[4] G. Nierhaus, *Algorithmic Composition: Paradigms of Automated Music Generation.* SpringerWienNewYork, 2009.

[5] M. Edwards, "Algorithmic composition: Computational thinking in music," *Communications of the ACM*, vol. 54, no. 7, pp. 58–67, 2011.

[6] D. Conklin, "Music generation from statistical models," in *Proceedings of the 2003 Symposium on Artificial Intelligence and Creativity in the Arts and Sciences*, 2003, pp. 30–35.

[7] F. P. Brooks, A. L. Hopkins, P. G. Neumann, and W. V. Wright, "An experiment in musical composition," *IRE Transactions on Electronic Computers*, vol. 5, pp. 175–182, 1957.

[8] W. Schulze and B. Van der Merwe, "Music generation with markov models," *IEEE Multimedia*, vol. 18, no. 3, pp. 78–85, 2011.

[9] I. Simon, D. Morris, and S. Basu, "Mysong: Automatic accompaniment generation for vocal melodies." in *Proceedings of the 2008 Conference of Human Factors in Computing Systems*. ACM Press, 2008, pp. 725–734.

[10] J. Paiement, D. Eck, and S. Bengio, "Probabilistic melodic harmonization," in *Proceedings of the 19th Canadian Conference on Artificial Intelligence*, ser. Lecture Notes in Computer Science, vol. 4013. Springer, 2006, pp. 218–229.

[11] M. Mohri, "Weighted automata algorithms," in *Handbook of Weighted Automata*. Springer, 2009, pp. 213–254.

[12] M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech and Language*, vol. 16, pp. 69–88, 2002.

[13] S. M. Katz, "Estimation of probabilities from sparse data for the language model of a speech recognizer," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 400–401, 1987.

[14] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, pp. 4–16, 1986.

[15] N. Jiang, "An analysis of automatic chord recognition procedures for music recordings," Master's thesis, Saarland University, 2011.

[16] J. Graehl, "Carmel," 2008, available at: http://www.isi.edu/licensed-sw/carmel.

[17] A. Sorensen and A. R. Brown, "Introducing jmusic," in *InterFACES: Proceedings of the Australasian Computer Music Conference*, 2000, pp. 68–76.

[18] F. Drewes and J. Högberg, "An algebra for tree-based music generation," in *Proceedings of the 2nd international conference on Algebraic informatics*, ser. Lecture Notes in Computer Science, vol. 4728. Springer, 2007, pp. 172–188.