THE CLOSURE-BASED CUEING MODEL: COGNITIVELY-INSPIRED LEARNING AND GENERATION OF MUSICAL SEQUENCES

James Maxwell
Simon Fraser University SCA/SIAT
jbmaxwel@sfu.ca

Philippe Pasquier
Simon Fraser University SIAT
pasquier@sfu.ca

Arne Eigenfeldt
Simon Fraser University SCA
arne_e@sfu.ca

ABSTRACT

In this paper we outline the Closure-based Cueing Model (CbCM), an algorithm for learning hierarchical musical structure from symbolic inputs. Inspired by perceptual and cognitive notions of *grouping*, *cueing*, and *chunking*, the model represents the *schematic* and *invariant* properties of musical patterns, in addition to learning explicit musical representations. Because the learned structure encodes the formal relationships between hierarchically related musical segments, as well as the within-segment transitions, it can be used for the generation of new musical material following principles of *recombinance*. The model is applied to learning melodic sequences, and is shown to generalize perceptual contour and invariance. We outline a few methods for generation from the CbCM, and demonstrate a particular method for generating ranked lists of plausible continuations from a given musical context.

1. INTRODUCTION

1.1 Music Perception: Specificity and Invariance

The music perception and cognition literature has long acknowledged the existence of categories of perceptual change used by listeners to build mental representations of musical forms. The term "contour" has been used to describe *directionality* in perceptual change [1, 2, 3] and studies have shown contour to be a primary attribute used in the short-term recognition of basic musical patterns, and the comprehension of musical structure [4, 5, 6]. At a more detailed level of description, there are also categories of *invariance*, which express the perceptual similarity of patterns which are quantitatively *dissimilar*. For example, the music descriptor "pitch interval" is invariant across changes of absolute pitch level, just as the category of "rhythm" is invariant across changes in tempo. Finally, at the most detailed level of description, there are quantitatively "identical" percepts—specific pitches, for example. These three levels of perception share a complex interaction during music listening, which appears to develop with age and musical experience (for an overview, see Dowling [7]).

1.2 Probability or Planning?

Ever since the publication of Shannon's "Information Theory", musicians and researchers have focused on the information-theoretic properties of musical structure (in particular, the "Markov property" [8]) when considering the type of formal continuity demonstrated by a given musical "style." Although the importance of such information-theoretic properties cannot be denied, we suggest that they are *effects* of musical thinking, not necessarily generative factors. For the CbCM, we propose an alternative approach founded on ideas from music perception, cognition, and memory. Although the CbCM demonstrates information-theoretic properties similar to other related models, it will be seen that the learned structure allows for an approach to generation unlike that applied in conventional Markov models.

Like our earlier MusicDB [9], the CbCM was designed as a 'musical memory', to be used in a larger interactive composition system. Aspects of the CbCM can be compared to Lartillot's models for motivic extraction [10, 11], but whereas those models focus on pattern detection, the CbCM emphasizes hierarchical structure and sequence generation. The CbCM also shares some similarity with Deutsch & Feroe's hierarchical pitch representation [12], though the formalism differs considerably. Following a detailed discussion of the model, we will give examples of its implementation in a composition environment called *ManuScore*, where it is used to generate sorted lists of phrase continuations from a given musical context.

The CbCM makes no claims of biological plausibility. Rather, it takes theoretical ideas from the cognitive science of music as jumping-off points in the formulation of a model for hierarchical sequence learning and generation.

1.3 Ideas and Terminology

The CbCM has been designed with reference to a number of principles of musical memory:

1. *Association*: The process by which events that occur in close temporal succession form connections in memory.
2. *Cueing:* The process through which "one memory *cues* another memory with which it has formed an association" [13].
3. *Closure*: "When some aspect of the acoustical environment changes sufficiently, a boundary is created"

[13]. The perception of this boundary is referred to as *closure*.

4. *Grouping*: "The tendency for individual items in perception to seem related and to bond together into units" [13]. In the context of music, events tend to be *grouped* around points of perceptual *closure*.

5. *Chunk:* "Chunks are small groups of elements (5-9) that, by being frequently associated with one another, form higher-level units, which themselves become elements in memory" [13].

1.4 Two Dimensions of Hierarchy

Hierarchical models of musical form, like Lerdahl & Jackendoff's "Generative Theory of Tonal Music" (GTTM) [14] organize musical materials into segments of increasing duration, building "motifs" at the lowest levels, "phrases" and "sections" at higher levels, and finally complete compositions at the highest levels.

However, there is another type of hierarchy in music which is perhaps not immediately apparent, and which has not generally been applied in computational models of music learning and generation; a dimension relating to the *specificity* of perceptions. When describing the pitch sequence {C4 G4}, many computational models will represent this sequence by enumerating its various *attributes*; the pitches C4 and G4, the melodic interval of 7 semitones, and perhaps the contour ("+"). This conception tends to give equal weight to each attribute in the music representation. We propose that the *invariant* categories of interval and contour are not merely *attributes* of an event, but rather represent hierarchically prerequisite states, such that each pitch in a sequential context *requires* an interval, and each interval *requires* a contour. Such a conception of hierarchy is similar to Conklin's notion of "subsumption" [15], though our implementation expresses this idea explicitly in the CbCM topology (see Figure 1). The primacy of contour in short-term melodic recognition and the adaptability of melodic recognition to changes in absolute pitch level [1, 6] support this general conception of hierarchy.

With this in mind, the CbCM employs a music representation with three hierarchical levels of specificity: 1) **Schema**, the fundamental level pertaining to the detection of perceptual change (i.e., *pitch contour*), 2) **Invariance**, which captures relative quantities like those described by *pitch intervals*, and 3) **Identity**, which deals with the absolute quantitative values of the percepts themselves. Note that the application of such concepts need not be limited to pitch material; "contour", for example, could be applied to rhythmic *augmentation* and *diminution*, or to changes in harmonic *tension* or *density*.

2. OVERVIEW OF THE MODEL

2.1 General Design

The CbCM can be conceptualized as a graph, the structure of which is learned from a time series of symbolic musical inputs. The graph is hierarchical and concurrent, so that states of the graph are represented by one or more active

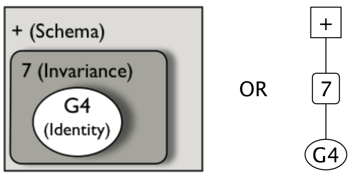

Figure 1. "Nesting" nodes of increasing specifity.

nodes. Concurrency in the CbCM reflects the manner in which musical structure unfolds simultaneously along multiple hierarchical dimensions. For example, a common statement like "*at the end of the third verse*" implies a location along three concurrent temporal dimensions: "at the **end**"—a position at the level of a musical *phrase*, "of the **third**"—a location in the complete form of the song, and "**verse**"—a position at the level of a musical *section*.

The states of the CbCM are ordered along two hierarchical dimensions; a *vertical* dimension, and a *nesting* dimension. The *vertical* dimension is explicitly sequential, and is arranged into one or more levels, each of which corresponds to a level of formal organization similar to those proposed by models like the GTTM. The *nesting* dimension organizes states and their substates according to the three levels of hierarchical specificity mentioned above: **Schema** , **Invariance**, and **Identity**. Considering these relationships under the formalism of Hierarchical State-Machines, we can say that **Identity** states are *implicitly* **Invariance** states, which are *implicitly* **Schema** states, as suggested by Figure 1.

For the sake of simplicity, we will use the term "levels" to refer to locations along the *vertical* dimension, and the term "states" (substates/superstates) to refer to locations along the *nesting* dimension.

2.2 Preprocessing and "Closure"

The CbCM builds its specific structure based on the notion of *closure*; i.e., the delineation of formal boundaries by the perception of musical change. However, it does not define or calculate the parameter over which change is detected. Rather, this value—referred to as the *closureSignal*—must be provided with the stream of input events. The CbCM detects significant changes in the *closureSignal*, and uses these changes to define segments in the learned model. This design is convenient, as it decouples the segmentation *criteria* from the hierarchical learning process; i.e., the specification of the *closureSignal* can be tailored to the type of input (melodic, rhythmic, harmonic, etc.).

A preprocessing step must be used to calculate the *closureSignal* for each input, in consideration of the type of input and the specification of an appropriate *closure* measure for that type. By isolating different input types in this manner, we acknowledge Snyder's notion of "soft closure" (though we will refer to this generally as *closure*). In our implementation, we are learning/generating melodic pitch sequences, and use calculations for melodic pitch expectancy based on Margulis [16], and rhythmic expectancy based, in part, on Desain [17]. Details of the preprocessing used in the current study are given in Section 4.2.

Computationally, the CbCM is realized as a network, the nodes of which are arranged into one or more levels, each of which contains a sequence of states (and their substates). Each node on a level has a single parent, and each substate has a single superstate, so that the sequential structure within a given level forms a tree, as does the state structure.

2.3 Learning and Inference in the CbCM

The CbCM is an online learner, and thus always runs its inference step *before* proceeding with learning. Each level of the CbCM has a single *rootNode* and a single *contextNode*. The *rootNode* does not store any information, but rather serves as an initial state from which edges to other nodes can be searched and/or created. The *contextNode* acts as a pointer to the current state on a given level. It is updated with each input, and thus progresses through its level as musical transitions are perceived. Each time a given node becomes the *contextNode*, a *counts* variable is updated, for use during generation. Because each level has its own *contextNode*, the model progresses through all levels *simultaneously*, reflecting the notion of concurrency described in Section 2.1.

Since the *contextNode* acts as a pointer to the current state of the CbCM, inference involves a search through the *contextNode's* attached edges for a transition matching the input. Failure to find a matching transition at the current state forces the search to be repeated at the superstate. Thus, failure to match the **Identity** state (*pitch*) forces a search of the **Invariance** state (*interval*), and failure to match the **Invariance** state forces a search of the **Schema** state (*contour*), as shown in Figure 2 (in the diagram, *contextNodes* are represented in white, light grey nodes represent superstates, and dark grey nodes represent hypothetical learned states). This pattern of defaulting toward the superstate when a transition cannot be found characterizes the *generalization* process used by the CbCM.

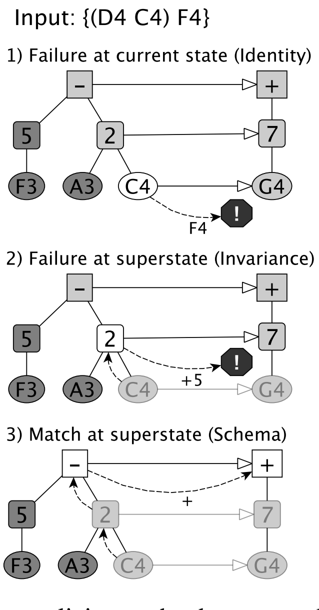
Input: {(D4 C4) F4}
1) Failure at current state (Identity)
F4
2) Failure at superstate (Invariance)
+5
3) Match at superstate (Schema)
+

Figure 2. Generalizing to the deepest applicable state.

Learning in the CbCM involves the construction of a hierarchical network representing the states and transitions embodied by a set of source works. At initialization, the CbCM has a single level with its *contextNode* set to the *rootNode*. Since only the *rootNode* node exists, the current state is assumed to be **Schema**. Learning cannot proceed without an initial context, so the first input is ignored. When processing subsequent inputs, the CbCM first performs inference, as described above. Since no transitions can be found, the model proceeds with learning. Learning a new transition involves adding a new node, expressing the greatest *specificity* possible, to the network. With only the *rootNode* in place, the CbCM extracts the **Schema** information from the input, creates a new **Schema** state node, and connects its edge to the *rootNode*.

The pattern for learning new nodes/states is shown in Figure 3. If the current state has no transitions to the input, only the **Schema** state can be learned, as shown in the 1st iteration. If an appropriate learned state can be reached via the current state's superstate, then the input can be added as a substate of the reached state, as in the 2nd iteration of Figure 3. This process continues until the **Identity** state representation has been learned (3rd iteration). In a trained CbCM, the edges connecting nodes represent *associations*, with the strongest associations being made between **Identity** states (since **Identity** states can only be formed through repeated exposure to a particular transition).

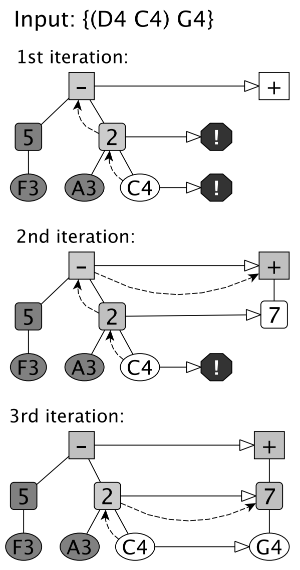

Figure 3. Learning states in order of reachability.

2.3.1 *Closure and CbCM Structure*

As mentioned in Section 2.2, the specific structure of the CbCM is determined through changes in *closure*, the value of which is calculated during preprocessing. The node itself stores the *closureSignal* value, which is updated each time the node becomes the level's *contextNode*. The input *closureSignal* is monitored at each time step, and moments of *decreasing* closure are used as indicators of formal change; i.e., when the *closureSignal* of the current input falls below that of the *contextNode*, a boundary is formed. During inference, this boundary forces the CbCM to limit its search for transitions to those edges with connections into *higher level* nodes. During learning, the boundary marks the end of the current segment, forcing the learned node to be added *to a higher level in the model*, as shown in Figure 4.

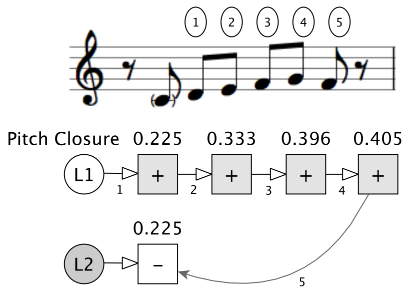

Figure 4. A decrease in *closureSignal* forces learning to a higher level.

Nodes created at higher levels thus represent *chunk boundaries* [13], marking the beginnings of perceptual segments. Each higher level node *cues* an **L1** node, so that the continuation of a higher-level node is *always a sequence of L1 nodes*. This can be seen in Figure 5, which shows the *cueing* connection (edge 6) made between the **L2** *contextNode* (**-**) and the new **L1** node (**-**). In the learning algorithm, this pattern is achieved by setting the level's *contextNode* back to the *rootNode* every time a segment boundary is detected. When the following E4 is received (Figure 5), the search at **L1** is carried out on the *rootNode* (which is now the *contextNode*). Since no descending **Schema** transition has been learned, a new node is added to the *rootNode*. At the end of a single pass through the input sequence in Figure 5, the model has learned two contour segments: {**+ + + +**} and {**- -**}.

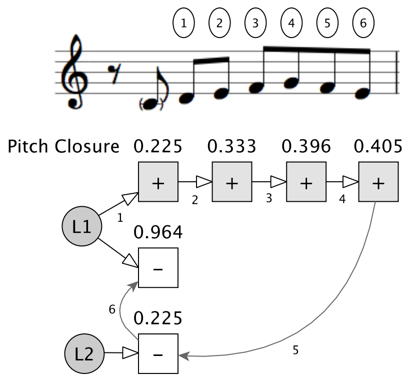

Figure 5. Learning reverts to **L1** after a higher-level transition.

When learning creates nodes at higher levels (i.e., beyond **L2**) the process follows the same general pattern. If **L1** detects a segment boundary, the CbCM attempts to learn the input at **L2**, but if **L2** *also* detects a segment boundary, learning is passed to **L3**, and so on.

In order to clarify the formal relationship between higher-level nodes, we make an additional across-level connection when adding nodes above **L2**. This connection passes from the *contextNode* of the current segment's level to the newly learned higher-level node. We refer to this connection as a "formal cue", since the transition it describes is never directly output during generation; it serves only to form a *cueing* relationship between *chunk boundaries*, thus establishing hierarchical structure.

A diagram of a pitch CbCM trained on *a single pass* of the opening theme from Bach's BWV 846 (Fugue) is given in Figure 6. In the diagram, the contour symbols (+, -) indicate **Schema** states, the round-bracketed numbers indicate **Invariance** substates, and the square-bracketed numbers indicate **Identity** substates. The curved arrows from **L1** indicate across-level *cues*, and the curved arrows connecting higher levels back into **L1** show the cueing function of *chunk boundaries*. The *formal cues* in the model are labelled with italics (F1, F2, and F3).

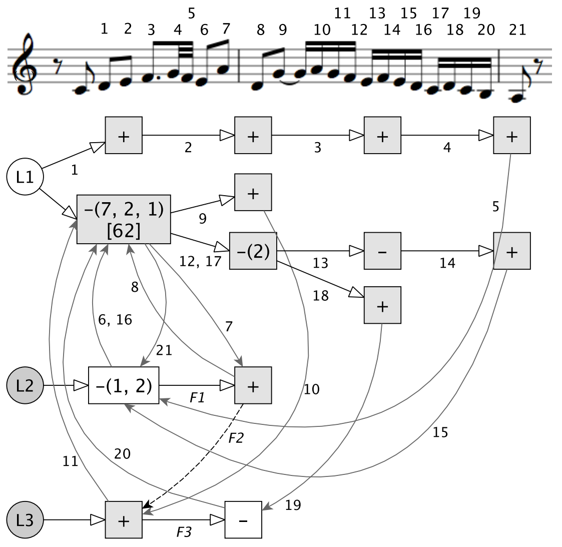

Figure 6. A CbCM for pitch, trained on a single pass of the theme from Bach's BWV 846 (Fugue).

2.3.2 A Note About Trained CbCM Structure

A few observations can be made about the CbCM, considering it as a graph—or rather, a composite of hierarchically-related graphs—in which the nodes at each state (**Schema, Invariance, Identity**) form *directed graphs* with the following properties:

1. For each level in the model a tree subgraph exists, describing the work/corpus at a particular level of temporal/formal organization.

2. The **L1** tree represents the set of all perceptually grounded segments in the corpus.

3. Each level above **L1** also represents a set perceptually grounded segments, but at higher level of formal structure (the segments contained by higher levels are analogous to *"time span reductions"* in the GTTM).

4. The **L1** tree is the intersection of all higher-level subgraphs in the model (i.e., all higher levels have paths through **L1**).

5. *Formal cues* ensure that all higher-level nodes also form a separate subgraph describing the relationship between all perceptual (**L1**) segments in the corpus.
6. For every higher-level node Ln_x there exists at least one path through **L1**, of $length > 1$, which terminates in an adjacent higher-level node Ln_y.
7. In a trained CbCM there exists at least one path ${}_xP_y$, in each state graph (**Schema, Invariance, Identity**), describing the complete sequence of transitions in a particular source work.

3. GENERATION FROM THE CBCM

The CbCM is a hierarchical encoding of musical information designed with an emphasis on sequence generation. One of our design priorities was to link the learning and generation processes in a cognitively grounded manner. To do this, we modelled the algorithms on ideas from Logan's "Instance Theory" of learning. Logan's theory proposes that novices initially approach a problem with a "general algorithm that is sufficient to perform the task", but that with repeated exposure they eventually learn to "respond with a solution from memory" [18]. The theory explains the iterative nature of learning and the rapid increases in efficiency observed with repeated exposure to the conditions of (and solutions to) a given problem.

Given the learning algorithm described in Section 2.3, we can see how the CbCM might demonstrate *instance-based* learning in the context of generation. Consider the problem of trying to repeat the example transition {C4 G4}. After a single exposure, the CbCM would extract only contour information: "**Schema** = +". When trying to repeat the transition, a "general algorithm" could thus proceed via heuristic search; i.e., using the **Schema** information to build a search space of pitches *above* C4. After a second exposure to the transition, the *interval* substate "**Invariance** = 7" would be learned, and generation could proceed according to a "rule"—i.e., 'add 7 to the previous note.' Finally, after a third exposure to the sequence, the CbCM would learn the substate "**Identity** = G4", and could recall the transition directly from memory.

Of course, when executing the "general algorithm", we must have some criteria for evaluating potential solutions in the search space. Here we turn to Ritchie's notion of "quality." Ritchie defines *quality* as a degree of membership in the set of objects that define a given "class"—a genre, for example [19]. Since the CbCM uses the *closureSignal* as a constraint on well-formedness, it follows that the calculated *closureSignal* of a given production will reflect the *quality* of that production. Our basic quality calculation is:

$$\begin{array}{c} Q_s = f(contextNode, s) \\ s \in S \end{array} \tag{1}$$

where Q_s is the quality rating of production s, $f()$ is a function used to calculate the *closureSignal* of a given transition (used during preprocessing), and S is the set of possible productions. The sequence $(contextNode, s)$ represents the transition from the *contextNode's* value (e.g., a MIDI note) to the value of a possible production (output) which, in this case, would also be a MIDI note.

In a trained CbCM, the *previously learned* transitions should provide 'exemplars' for good productions, so that the highest quality *novel* productions should result in *closureSignals* proximal to those produced by the learned transitions:

$$\begin{array}{c} Q_n = 1 - \underset{\{\tau \in T\}}{\min}(|Q_n - Q_\tau|) \\ n \in N \end{array} \tag{2}$$

$$N = S \setminus T \tag{3}$$

where Q_n is the quality rating of production n, N is the set of *novel* productions at the *contextNode*, and T is the set of productions made possible by the learned transitions at the *contextNode*. Depending on the input type, S may be infinitely large, as is the case with 'unquantized' rhythmic values. Thus, for practical purposes, we limit S to some finite set of discrete symbols—i.e., quantized rhythmic values or MIDI note numbers.

3.1 Generation by Planning

From the preceding discussion, it is clear that two basic approaches to generation are possible: 1) selecting transitions based on *quality*, and 2) selecting transitions probabilistically using the *counts* values of all reachable nodes. The first approach will be strongly influenced by the *closureSignal* calculation function, while the second approach will result in behaviour analogous to a variable-order Markov model. Of course, a combination of *quality* and probability could also be used.

However, the CbCM also provides a mechanism for generating segments through a process of *planning*. This is the approach used for generating continuations in our *ManuScore* composition environment, discussed in Section 4. In this approach, we consider transitions not at the note-to-note level, but rather at the phrase level. If we look at Figure 6 from the perspective of **L2**, we can think of **L2-Node1(-2)** as a *goal*, which can be successfully achieved by a specific *plan*—in this case, the **L1** sequence {**+ + + +**}. This segment can be generated by backtracking through edge 5 to **L1-Node4**, and along the chain of *parent* nodes, until we reach the *rootNode*, as shown in Figure 7.

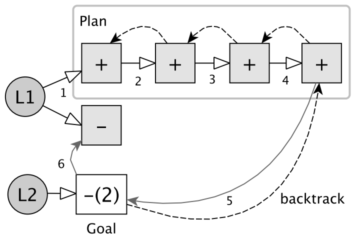

Figure 7. Building a *plan* through backtracking.

Generating plans from levels above **L2** follows an "unwinding" pattern, in which plans alternate between *output plans* (sequences of **L1** nodes) and *formal plans* (sequences of higher-level nodes). The *formal plans* are determined by backtracking along the *formal cues* discussed in section 2.3.1. Since *formal plans* are sequences of higher-level nodes, they do not result in output, but rather provide *chunk boundaries* from which we can generate *output plans* (which can be directly output). As plans are unwound from the top, each generated *output plan* can be placed on a stack for subsequent evaluation.

3.2 Novelty and Quality in the CbCM

Ritchie defines *novelty* as the degree of *dissimilarity* of a production to existing examples of that genre [19]. Assuming that the CbCM has been sufficiently trained on appropriate examples (that is, its structure represents a "genre"), we can evaluate *novelty* in terms of the intersection between a given subgraph in the CbCM and the inferred subgraph of the production. For a given state/node, Equation 3 defines a *local* set of novel productions (transitions not yet learned in the current context) with which a more general measure of novelty can be calculated.

Of course, producing a single novel transition doesn't guarantee "novelty" any more than exploiting a known transition prohibits it. Novelty is dynamic and cumulative. For this reason, it is useful to calculate the *potential* for novel generation at the current time step ($NLim^t$) cumulatively through time:

$$NLim^t = \frac{\sum_{t=0}^{n} \left(\frac{|N|}{|S|}\right)^{t-n}}{n+1} \tag{4}$$

Equation 4 can be used to determine $NLim^t$ over the entire CbCM, by continually incrementing n from the beginning of training/inference, or it can be used locally, by setting n to the sequential position of the *contextNode* on its branch. The *novelty rating* Λ_p^t of a given production p will thus be:

$$\Lambda_p^t = \begin{cases} NLim^t & \text{if } p \in N \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

The use of $NLim^t$ can allow the model to determine the probability of generating a novel production in the current context, while Λ_p^t helps estimate the novelty of a given production from a trained CbCM.

4. IMPLEMENTATION

As mentioned above, we designed the CbCM to serve as a musical memory for a larger music composition system. In our *ManuScore* software, the user can request continuations of a given context, for which the CbCM provides a sorted list of possible options. The user can toggle through the ranked pitch and rhythm segments independently, auditioning each pitch/rhythm combination via MIDI.

4.1 Pitch and Rhythm Models

In order to model the phenomenon of *primitive grouping*—the independent grouping of pitch and rhythm information at low levels of auditory processing [13]—while at the same time modelling the interaction of pitch and rhythm in music cognition, we designed our melodic learning/generation system to utilize two CbCMs; one for pitch and another for rhythm. Each model performs "primitive" segmentation at **L1**, by monitoring changes of *closure* in a single domain, but both models combine pitch and rhythmic closure information when segmenting at higher levels. In this way, the trained system retains a vocabulary of independent pitch and rhythm 'motives' at **L1** (which can be combined in novel ways during generation), but is also able to represent high-level musical form. During generation, we construct *plans* for each model independently, then pair up the pitch and rhythm plans for rendering to the score.

4.2 Preprocessing of the *closureSignal*

In our implementation, a preprocessing step determines the *closureSignal* for pitch using Margulis' "Melodic Expectancy Model" [16]. We *do not* apply the model in its complete form, but use only the "basic expectancy" calculation, treating points of *decreasing* expectancy as instances of *"soft closure"* [13]. Because we are not modelling harmonic context in the current study, we omit the "stability" coefficient from Margulis' formula (as recommended in [16]):

$$z = (p \times m) + d \tag{6}$$

where z is the expectancy, p is the "proximity rating", m is the "mobility" (as in Margulis', 2/3 for repetitions, 1 otherwise), and d is the "direction rating." For details see Margulis [16]. We scale z to real a number in the range [0,1] and use it as our *closureSignal* for pitch.

To calculate rhythmic closure, we use a combination of rhythmic proximity and rhythmic expectancy. For the expectancy calculation we use Desain's "basic expectancy" from his "(De)composable Theory of Rhythm" [17]. For the proximity calculation we use a simple exponential function:

$$p = 1.0 - \frac{x^2}{n^2} \tag{7}$$

where p is the calculated proximity, x is the IOI time, and n determines the temporal window over which connectivity will be maintained—we set this value to 6000 milliseconds to approximate the span of short-term memory [13]. Because any duration of silence separating events decreases their perceptual connectivity [13], we apply a further scaling for events separated by a silence greater than 800 milliseconds.

$$p = \mu - \frac{\mu \times (x - 800)^2}{n^2} \tag{8}$$

where μ determines the initial scaling amount, x is the separation time, and n determines the temporal window over which the scaling will occur (5200 ms in our model). We set μ to 0.7, so that any separation greater than 800 ms incurs an immediate 70% reduction in connectivity. If $p < 0$ we set it to zero.

The final rhythmic *closureSignal* z is the product of the proximity and expectancy:

$$z = p \times s \tag{9}$$

5. MODEL TRAINING AND TESTING

We trained *ManuScore* on a monophonic arrangement of the Fugue from Bach's BWV 846, from the Well-Tempered Klavier. Our choice to limit the source material to a single work allowed us to easily identify novel productions. In making the arrangement we tried to maintain as much of the work's formal integrity as possible, while reducing the four-part setting to a single monophonic voice [1] .

To test both inference and generation, we generated output phrases as *continuations* of a given context. A melodic fragment was entered by hand, and the CbCM performed inference on the fragment and updated its state accordingly. Generation was then initiated from the **L2** *contextNode* of both models, and segments were formed via backtracking (Section 3.1). The backtracking process halted when the inferred state was reached; i.e., we backtracked to the point where the input fragment ended. All segments generated in this manner were sorted by summing the *counts* values of their constituent nodes. When determining the *count* attributed to a given state/node, we summed the *counts* values of all implicit superstates (i.e., for an **Identity** plan node, we included the *counts* of its **Invariance** and **Schema** superstates).

6. RESULTS

As an initial test, we entered the first two pitches of the training work, which produced the continuation in Figure 8-A. Here, the top staff represents the input context and the bottom staff represents the CbCM's continuation. The greyed-out note (F4) in the continuation indicates that the rhythmic value was not part of a generated plan, but rather was determined probabilistically in order to provide IOI values for all notes in the generated pitch sequence [2] . The label on the generated segment "P 1/7 - R 1/2" indicates that the CbCM is displaying the first of seven pitch options combined with the first of 2 rhythm options. Although the CbCM in *ManuScore* does infer/generate rhythmic IOI values, it does not handle note durations, so we extended the 'sustain bars' in the generated output by hand, for the sake of clarity. Toggling to continuation "P 2/7 - R 2/2" we get the pitch/rhythm combination shown in Figure 8-B, which is a *novel* sequence not contained in the source work.

In Figure 8-C, we transposed the context fragment up a tritone to {F#4 G#4}, producing a *context* not found in the source work. The output is essentially the same, generating 7 pitch options and 2 rhythm options, but has been transposed up a tritone. This spontaneous transposition indicates that the CbCM has 'defaulted' to the **Invariance** state during inference and modified its continuations accordingly. A similar test is shown in 8-D, except that in this case we entered a 4-note fragment which followed the *contour* of the original, but not the interval sequence. The production correctly completes the contour by continuing with a {**+2 -2**} pattern. Inspection of the pitch CbCM during inference revealed that the transitions {C4 F#4} and {F#4 B4} from the context were inferred at the **Schema** state, as expected, while the transition {B4 C5} was inferred at the **Invariance** state. Since the source work has a "+1" transition {E4 F4} in the same context, this behaviour was also expected, indicating that the CbCM was able to transition to a more *specific* substate.

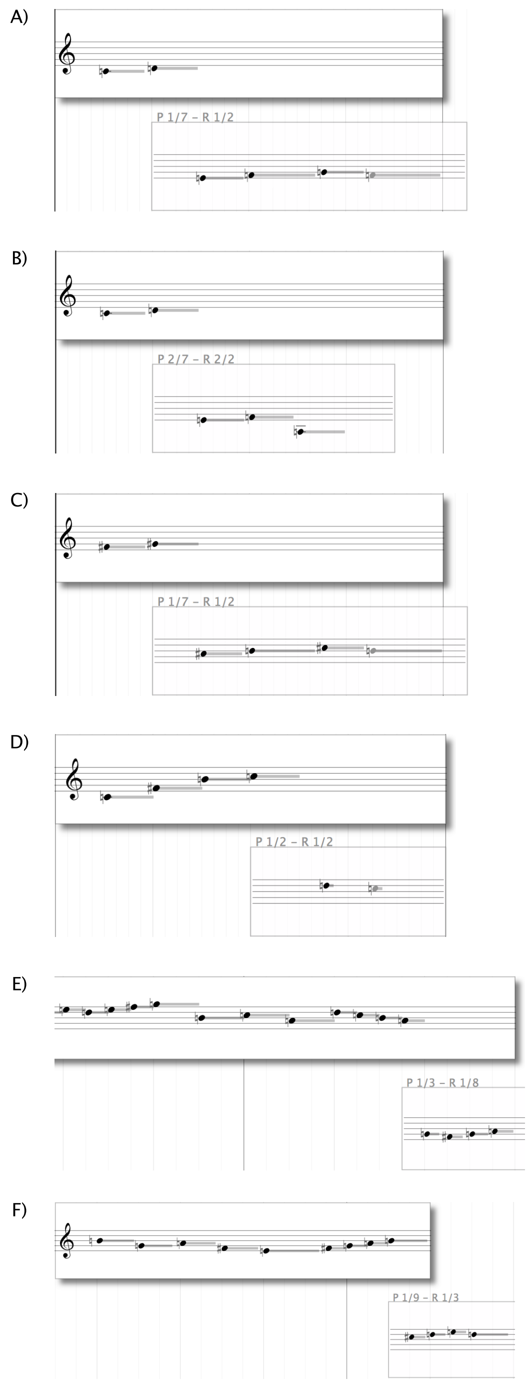

Figure 8. Continuations generated by the CbCM in *ManuScore*.

For our next test, we entered a longer context running from the beginning of the source work to half-way through measure 5. The first option produced from this longer con-

[1] The arrangement can be viewed at: `http://rubato-music.com/home/Media/Bach_846_Fugue_monophonic.pdf`

[2] It is worth note that this IOI is incorrect; in the source work, the F4 falls on the downbeat immediately following the G4.

text is shown in Figure 8-E. The sequence produced, {G4 F#4 G4 A4}, is once again a novel segment, which does not appear in the original. It provides a valid continuation of the context, and is of particular interest since it maintains the modulation to the dominant (alteration of F4 to F#4) introduced in the previous measure. Option "P 3/3 - R 3/8", produced from the same context, produces a quotation of the original.

Finally, we entered a completely novel, but still idiomatic context fragment, resulting in the production shown in Figure 8-F. The continuation is stylistically appropriate and is not a direct quotation from the source work.

7. CONCLUSION

The continuations generated by the CbCM in this preliminary test suggest that the model may be capable of generating context-sensitive "quotations" from the training set, in addition to reasonably well-formed novel productions.

The training process showed the expected pattern of learning, achieving complete training (i.e., learning all **Identity** states, as discussed in Section 2.3) after 3 passes over the source work. After training, both the pitch and rhythm CbCMs created 4 hierarchical levels. The pitch model created 181 **Identity** state nodes, 68 **Invariance** nodes, and 35 **Schema** nodes, over 274 transitions. Because the pitches comprising the source work are represented only by the **Identity** nodes (**Invariance** and **Schema** states are *implicit*), we use the count of **Identity** nodes for the calculation of compression, resulting in a 66% compression ratio.

8. FUTURE WORK

Our next step with the CbCM will be to evaluate performance with a larger body of source works. We are also interested in examining more closely the segmentation produced by our melodic expectancy-based approach. Although melodic expectancy seems a reasonable candidate for a *closureSignal*, the determination of an ideal *closure* calculation remains an open question. There is also a great deal of room for investigation into the various methods for generation offered by the structure of the CbCM as a memory model. We are currently developing a modular cognitive architecture for music, using the CbCM as a form of long-term memory.

Acknowledgments

This research was made possible, in part, by a grant from the *Social Sciences and Humanities Research Council of Canada*.

9. REFERENCES

[1] W. Dowling, "Context effects on melody recognition: Scale-step versus interval representations," *Music Perception*, vol. 3, no. 3, pp. 281–296, 1986.

[2] D. Deutsch, *The psychology of music*. Academic Pr, 1999.

[3] D. Levitin, "Memory for musical attributes," *Foundations of cognitive psychology: Core readings*, pp. 295–310, 2002.

[4] W. Dowling and J. Bartlett, "The importance of interval information in longterm memory for melodies," *Psychomusicology: Music, Mind and Brain*, vol. 1, no. 1, 2008.

[5] A. Lamont and N. Dibben, "Motivic structure and the perception of similarity," *Music Perception*, vol. 18, no. 3, pp. 245–274, 2001.

[6] J. Edworthy, "Interval and contour in melody processing," *Music Perception*, vol. 2, no. 3, pp. 375–388, 1985.

[7] W. Dowling, *The development of music perception and cognition*. Foundations of Cognitive Psychology. Cambridge: MIT Press, 1999.

[8] C. Ames, "The markov process as a compositional model: a survey and tutorial," *Leonardo*, vol. 22, no. 2, pp. 175–187, 1989.

[9] J. Maxwell and A. Eigenfeldt, "The musicdb: A music database query system for recombinance-based composition in max/msp," in *Proceedings of the 2008 International Computer Music Conference*, 2008.

[10] O. Lartillot, "A musical pattern discovery system founded on a modeling of listening strategies," *Computer Music Journal*, vol. 28, no. 3, pp. 53–67, 2004.

[11] O. Lartillot and P. Toiviainen, "Motivic matching strategies for automated pattern extraction," *Musicae Scientiae*, vol. 11, no. 1 suppl, p. 281, 2007.

[12] D. Deutsch and J. Feroe, "The internal representation of pitch sequences in tonal music." *Psychological Review*, vol. 88, no. 6, p. 503, 1981.

[13] B. Snyder, *Music and memory: an introduction*. The MIT Press, 2000.

[14] F. Lerdahl, R. Jackendoff, and R. Jackendoff, *A generative theory of tonal music*. The MIT Press, 1996.

[15] D. Conklin and M. Bergeron, "Feature set patterns in music," *Computer Music Journal*, vol. 32, no. 1, pp. 60–70, 2008.

[16] E. Margulis, "A model of melodic expectation," *Music Perception*, vol. 22, no. 4, pp. 663–713, 2005.

[17] P. Desain, "A (de) composable theory of rhythm perception," *Music Perception*, vol. 9, no. 4, pp. 439–454, 1992.

[18] G. Logan, "Toward an instance theory of automatization," *Psychological review*, vol. 95, no. 4, pp. 492–527, 1988.

[19] G. Ritchie, "Assessing creativity," *Institute for Communicating and Collaborative Systems*, 2001.