MUSICOG: A COGNITIVE ARCHITECTURE FOR MUSIC LEARNING AND GENERATION

James B. Maxwell
Simon Fraser University
jbmaxwel@sfu.ca

Arne Eigenfeldt, Philippe Pasquier, Nicolas Gonzalez Thomas
MAMAS Lab, Simon Fraser University
mamas-lab@sfu.ca

ABSTRACT

Music composition is an intellectually demanding human activity that engages a wide range of cognitive faculties. Although several domain-general integrated cognitive architectures (ICAs) exist—ACT-R, Soar, Icarus, etc.—the use of integrated models for solving musical problems remains virtually unexplored. In designing MusiCOG, we wanted to bring forward ideas from our previous work, combine these with principles from the fields of music perception and cognition and ICA design, and bring these elements together in an initial attempt at an integrated model. Here we provide an introduction to MusiCOG, outline the operation of its various modules, and share some initial musical results.

1. INTRODUCTION

In their two-part survey of cognitive approaches to music computation, Purwins et at. [1, 2] highlight the important role that cognitive modelling has played in the field. However, they also underscore the general absence in music computation research of integrated approaches like those central to the field of ICA design [3, 4]. Rather than addressing the overall problem of modeling intelligent musical behaviour, research programs have tended to focus on isolated functions like key estimation, beat induction, voice-separation, melodic segmentation, and so on. Conversely, in the field of ICA design, models like ACT [5], Soar [6], and Icarus [7] have taken a 'top-down' approach, first looking at the roles that perception, learning, memory, and action might take in supporting human-level intelligence, then designing modular architectures integrating these primary functions, and testing their performance in domain-general tasks. In such research, the focus is on the theory embodied by the *architecture*, and the validity of the theory is judged on its performance in solving real world problems, or replicating results from the experimental literature.

Of course, in designing a cognitive architecture specifically for music, we are not attempting to model intelligence *in general*. Rather, we are proposing a 'top-down' approach to solving the 'problem' of music generation, using an architecture of interconnected modules.

2. MUSIC PERCEPTION & COGNITION

The field of music perception and cognition research is immense, and deeply interdisciplinary, covering areas ranging from the study of musical creativity in children, to the detailed neurobiology of music processing in the brain. In our work, we are particularly interested in the question of musical memory, and in how the mental representation of musical knowledge can support the specialized task of music composition.

In his book *Music and Memory* [8], Bob Synder provides an overview of the field. Of particular interest for our work are the principles of *association*, *cueing*, *grouping*, and *chunking*, and the manner in which these functions support the efficient use of mental resources. *Association* is the process through which events that occur in close temporal succession, or are related in some way, form connections in memory. *Cueing* is the process by which "activating one of the associated memories may also activate another memory with which it has formed an association" [8]. *Grouping* and *chunking* are processes through which singular memories are combined into higher level concepts. Specifically, grouping occurs when "some aspect of the auditory environment changes sufficiently" [8], and a perceptual boundary is formed. Chunking is the process through which groupings of single pitches are associated into "motifs", motifs associated into "phrases", and so on. Chunking allows these groupings to form singular mental concepts, optimizing the limited capacity of "short-term" memory through hierarchical nesting.

Synder also outlines three general levels of memory: *echoic memory*, *Short-Term Memory* (STM), and *Long-Term Memory* (LTM). Echoic memory operates on a very short time-scale, and is essential for low-level perceptual processes like onset detection, pitch perception, and so on. STM operates on a slightly longer time-scale—from 3 to 5 seconds (or 7 ± 2 discrete items)—and is essential for higher cognitive functioning. For our work we prefer to think in terms of the more general notion of "Working Memory" [9–11], which accommodates a broader conception of how short-term storage might function in a music learning/generation system. Finally, LTM represents the conventional notion of human knowledge as a semi-permanent store of information acquired through learning and experience.

3. INTEGRATED COGNITIVE ARCHITECTURES

One of the primary goals of ICA design is to model intelligence at the *architectural* level, as a general, system-level theory. Langley et al provide an overview of the main ideas and themes in the field, and outline four important systems: ACT-R, Soar, Icarus, and Prodigy [12].

Such architectures generally have a core set of functions in common: 1) Recognition, 2) Decision making, 3) Perception, 4) Prediction, 5) Problem solving and Planning, 6) Reasoning, 7) Acting, and 8) Learning. Different ICAs have different policies for handling these functions, and may attribute different degrees of importance to each one. In some cases, a given function may be represented explicitly, as a subroutine or module, while in others it may be implicit in the design of the knowledge representation (e.g., an *episodic memory* system may be considered implicilty *predictive*). One of the oldest cognitive architectures, ACT-R, includes sensory modules (Perception), motor modules (Action), an intentional module (Decision making, Problem solving, Planning, Reasoning), and a declarative module, or LTM (Recognition, Learning). Although the architecture doesn't possess an explicit STM, each module has its own 'buffer', which acts as a short-term store for relational knowledge structures (referred to as 'chunks').

Although the architectural details and implementation strategies vary, this basic process of perceiving the environment, weighing possible actions against past experience, and choosing actions to help achieve goals, is common to all ICAs.

4. MUSICOG: AN INTEGRATED MODEL

Just as the integrated cognitive systems discussed in Section 3 are modular in design, so too is MusiCOG divided into a set of processing modules, which work in conjunction when carrying out perceptual/cognitive tasks. An overview of the MusiCOG design is shown in Figure 1.

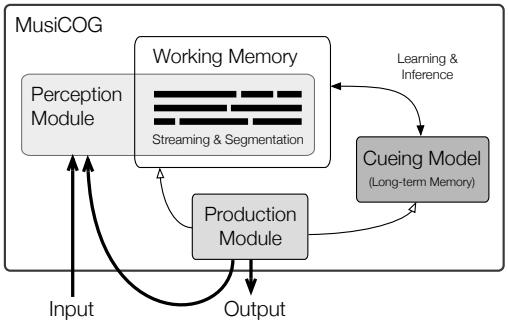

Figure 1. An overview of the MusiCOG architecture.

4.1 Processing Modules

We will start by introducing the basic processing modules in MusiCOG and outlining the music representations used when processing musical input.

Perception Module PE: Responsible for accepting musical input, separating polyphonic voices into *streams*, and performing low-level melodic segmentation on each *stream*.

Working Memory WM: A temporary memory for musical input, responsible for chunking familiar musical patterns and maintaining the set of active *streams*. In Figure 1, it will be noted that there is a considerable overlap between the PE and the WM, indicating the high degree of interaction between the two modules, particularly when handling processes of *streaming* and *segmentation*.

Cueing Model CM: A Long-Term Memory that learns the hierarchical structure of monophonic musical lines. The CM can learn from multiple, concurrent parts, and can build associations between individual parts in polyphonic textures. Formally, it is a hierarchy of linked graphs, each of which represents a different level of musical structure, similar to "time-span reductions" in Lerdahl and Jackendoffs "Generative Theory of Tonal Music" (GTTM) [13]. The CM is a revised version of our previous "Closure-based Cueing Model" (CbCM) [14]. As in the CbCM, each level in the CM can represent information using nodes at three different "degrees" of musical specificity. The zeroth degree represents **Schema** information, like pitch contour. The first degree represents musical **Invariance**, such as pitch *interval*, and the second degree represents **Identity** information; e.g., MIDI note number. Degrees are hierarchically dependent, so that a given **Identity** node will implicitly reference a single **Invariance** node, which will implicity reference a single **Schema** node.

Production Module PM: Responsible for the generation of musical output. During the generation process, the PM draws on the corpus-based knowledge of musical structure learned by the CM, and also on the local, cognitively salient material held in WM.

4.2 Music Descriptors

MusiCOG uses the conventional descriptors for symbolic pitch (C4 = middle-C), pitch interval (distance in semitones), and pitch contour (+, 0, -). For rhythm, we represent the absolute timestamps of events in real-numbered beats, such that whole numbers indicate beats and fractional values indicate offsets within the beat. We use the Inter-Onset Interval (IOI), in ms, to represent rhythmic **Identity** values. For rhythmic contour (**Schema**), we use the notation described above, and for **Invariance** we use a novel representation called the "Beat Entry-Delay" (bED). The bED indicates the duration in beats of an onset from the local beat of the *preceding* onset. Figure 2 gives an example of the bED representation of a simple rhythmic pattern. For events 1, 5, and 6, $bED = 1.0$ indicates that the event falls direclty on the beat, whereas event 2, $bED = 0.5$ is a fractional position. For fractional values where $bED > 1$, as with event 3, the integer part indicates the number of beats since the last detected beat. The function for deriving bED_t from two consecutive timestamps (n_{t-1}, n_t) is given in Equation 1. In a generative context, when given the previous timestamp n_{t-1} and bED_t, a new timestamp can be calculated using Equation 2.

Since the bED representation encodes onset times with

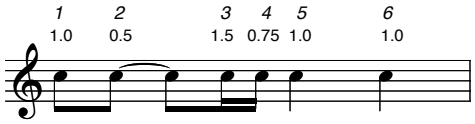

Figure 2. The Beat Entry-Delay representation.

reference to a local beat, it is particularly convenient for generation, as it allows generated rhythms to automatically align themselves with the beat structure of the new musical context. It is our feeling that the *bED* mirrors the cognitive reality of listeners more naturally than conventional onset interval-based (or ratio-based) representations.

$$bED_t = (n_t - \lfloor n_t \rfloor) + (\lfloor n_t \rfloor - \lfloor n_{t-1} \rfloor) \quad (1)$$

$$n_t = \begin{cases} \lfloor n_{t-1} \rfloor + bED_t + 1 & \text{if } bED_t \leq (n_t - \lfloor n_t \rfloor) \\ \lfloor n_{t-1} \rfloor + bED_t & \text{otherwise} \end{cases} \quad (2)$$

5. MUSIC PROCESSING IN MUSICOG

MusiCOG is best understood as an agent that listens to music, retains certain features of the musical surface, and gradually learns how the music is structured. When music has been learned, the agent can draw on the learned structure to generate novel musical forms.

5.1 The Perception Module (PE)

At initialization, the PE contains an empty set of *streams* $\Omega = \emptyset$. When MIDI inputs are received, they are collected into a new "group" G_t, where all events in the group have approximately the same onset time ($\pm$30 ms). With no active *streams*, the PE creates a new *stream* for each event. For future inputs, when $|\Omega| > 0$, events in G_t are assigned to *streams* using a cost-based voice-separation algorithm similar to other gestalt-based approaches [15, 16]. The PE's voice-leading cost calculation includes measures for pitch proximity, rhythmic proximity, melodic well-formedness, and avoidance of voice-crossings.

Once the events in G_t have been assigned to streams, the PE attempts to locate low-level segment boundaries for each active stream. Because we are modelling the notion of 'online' perception, we cannot rely on the use of a look-ahead function, as is common in melodic segmentation algorithms (for an overview, see [17]). However, due to the integrated nature of MusiCOG, we are able to include an influence from predictability, based on CM recognition (see Section 5.3). Taking this into account, we found that by using the inverse of the voice-leading cost, the predictability, and a small influence from pitch contour change [18], acceptable segmentation could be achieved.

When PE processing is complete, the incoming events from *group* G_t will be separated into $n > 0$ *streams*, and 'bottom-up', event-level *segment* boundaries will be defined.

5.2 The Working Memory (WM)

The WM is a temporary buffer for musical *elements*. These elements are stored in WM as hierarchical lists of events (i.e., notes), which represent either "segments" or "chunks." Segments can be single events, or sequences of chunks, whereas chunks are essentially 'wrappers' around segments. Elements are retained in WM based on a combination of the current WM capacity, element recency, and element cognitive salience. Capacity and recency are approximated using two parameters, n and m, which act as thresholds on the number of elements stored, and the allowable age of any given element, respectively. Element recency is taken as the timestamp of the *last* event in a given segment or chunk. Cognitive salience is a function of the degree of "musical parallelism" [13, 19] a given element shares with the current contents of WM; i.e., how many musically similar elements are currently being stored. The estimation of similarity is a complex task [19–21], which admittedly deserves closer attention. But as a provisional definition, we say that two elements are similar if they share the same **Schema** representation (e.g., matching contour). The total set of similar elements is thus analogous to Deliege's notion of the "imprint" [22].

Rather than increasing the cognitive salience of elements as parallelisms are detected, we instead use parallelism as a means to *prevent decay* of the cognitive salience. Each element is initially assigned a cognitive salience value based on CM recognition. This value will decay as a linear function of time. However, if a given stored element lacks parallelisms with the contents of WM, the rate of decay is accelerated, reducing the element's longevity in WM. When the number of stored elements exceeds the WM's capacity n, any element with a recency greater than m seconds and a cognitive salience less than a given threshold value, will be discarded (in our implementation, $n = 9$, and $m = 5$). Since the duration of retention is a function of the musical content, there is no *a priori* maximum retention time.

5.2.1 Chunking in the WM

When a new stream S is created, an empty segment is added to the end of the stream. As the PE assigns events to streams, each new event is appended to the end of the segment at S_1. When the PE detects a low-level segment boundary, a new segment is created for the boundary event, and inserted at S_1. The WM then "closes" the segment at S_2, indicating that it is complete.

At each time-step, the WM passes the segment at S_1 to the CM for learning/inference. When a segment is recognized by the CM, it is "wrapped" into a new chunk, which replaces the segment in WM. For each segment v^L (where L indicates the segment's hierarchical level) that is converted to a chunk $C(v^L)$ in this manner, WM capacity is increased by $|v^L| - 1$ memory spaces. Segments that are not recognized will eventually be discarded from WM. The calculation of segment/chunk recognition is based on the CM's ability to infer transitions ($\perp = 0$, **Schema** $= 0.3$, **Invariance** $= 0.8$, and **Identity** $= 1.0$). The final recognition level is the mean recognition of all inferred transitions in the segment/chunk.

5.2.2 Phrase Boundary Detection in the WM

The WM is also responsible for higher-level *phrase* segmentation, which is generally thought to be a 'top-down', cognitive process [23]. In MusiCOG, phrase boundaries are associated with points of musical parallelism—i.e., with the repetition of motivic patterns. Looking at the melody from Mozart's 40th Symphony (Figure 3), we can see two distinct phrases, the second of which begins one scale-step below the first. The asterisk marks the point at which the phrase boundary is identified.

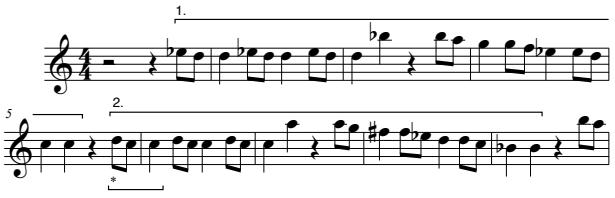

Figure 3. Parallelism in Mozart's 40th Symphony.

The WM finds these parallelism-based boundaries by looking for similarities between a newly formed chunk $C(v^L)$ and the older chunks stored in WM. When a familiar chunk is perceived in a larger phrase structure, this segment can serve as a phrase boundary [23]. To find such boundaries, the WM iterates over the contents of stream S, starting with the oldest element S_k, and searches for a parallelism between $C(v^L)$ and a preceding *chunk* ε. If a parallelism is found, a new 'chunk-segment' is started, to which all contiguous, intervening *chunks* are added, until $C(v^L)$ is reached, or the formal level of ε changes. This approach can be used for any level beyond L_1, allowing for the detection of parallelism-based boundaries at arbitrarily high levels of form. The only limit is the capacity of WM, which is defined primarily by the 'chunkability' of the music.

5.3 The Cueing Model (CM)

The CM represents MusiCOG's Long-Term Memory. It is a multi-dimensional, hierarchical graph, which is trained on segments from WM, and is used to make inferences on the contents of streams. It is an online learner, which learns to approximate the formal structure of a corpus of training works.

In contrast to the CbCM, which encodes surface events at all levels, in the CM, events are encoded at only L_1 and L_2: L_1 encodes short segments comparable to "motifs", whereas L_2 encodes sequences of motifs, to form "phrases" (see Figure 4). All higher levels encode sequences of paths from their adjacent *sub-levels*, so that L_3 encodes sequences of L_2 paths, L_4 encodes sequences of L_3 paths, and so on.

As mentioned in Section 4.1, the CM encodes events at different levels of musical specificity, referred to as "degrees" (called "states" in the CbCM—see [14]). Figure 5 shows all three degrees of the L_1 graph created by the two opening phrases from Mozart's 40th Symphony. It can be seen that each degree forms a separate directed graph, at a different level of specificity. It is worth noting that, although the **Schema** graph will always be a tree, the higher-

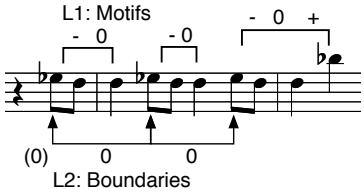

Figure 4. Encoding phrases as a combination of Motifs (L_1) and Boundaries (L_2).

degree graphs will not, allowing for increased compression.

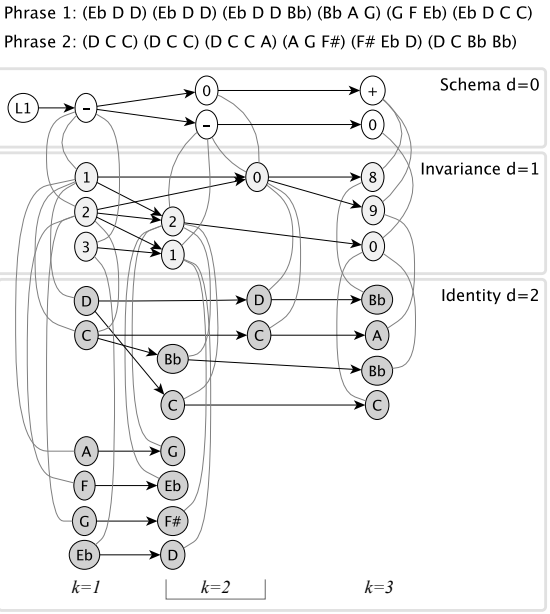

Figure 5. L1 of a CM trained on the opening of Mozart's 40th Symphony, showing all three degrees.

The CM produces fundamentally the same hierarchical structure as the CbCM, but the connectivity of the graph reveals significant differences. In our previous approach, we attemtped to learn the entire structure incrementally. This meant that across-level connections encoded the relationships between lower-level segment *endings* (i.e., "terminal nodes") and higher-level segment boundaries. However, with the addition of the WM, we are able to learn directly from sequences of *chunks*, and are thus able to build higher-level sequences somewhat independently of the lower-level sequential structure.

The difference between the two approaches is illustrated in Figure 6. Here we see that, due to the incremental learning pattern, the CbCM encodes $+1$ transitions at each L_2 node, representing the relationtionship between the D that ends each segment and its subsequent boundary E♭. In contrast, the CM encodes only the relationships between boundaries (i.e., interval 0). Thus, although the L_2 **Identity** information is the same (i.e., sequences of E♭s), the encoded **Schema** and **Invariance** relationships are not. We believe the CM's encoding to be a vast improvement, since it more accurately captures phrase-level structure and is

thus better able to generalize to different musical situations. For example, a CM trained on the first two complete phrases of the Mozart melody, as illustrated in Figure 7, will infer the inherent similarity of *all* phrases built from the direct repetition of L_1 segments/motifs, whereas the CbCM will not.

Figure 7 also shows the *links* connecting nodes on different levels. The dotted lines indicate *boundary* links, and the arrows between levels indicate *cueing* links. Boundary links are used to connect segment boundaries to lower-level terminal nodes, whereas cueing links connect terminal nodes to *subsequent* boundaries. In Figure 6 it will be noticed that whereas boundary links in the CM connect to terminal nodes, the CbCM always connects boundary links into depth $k = 1$—a consequence of the incremental learning process. The CM is able to make these more specific boundary links because it learns from complete chunks (i.e, with a boundary and terminal) at higher levels. Also note that whereas the CbCM always omits the first event (the initial E♭), the CM encodes *all* events on the musical surface.

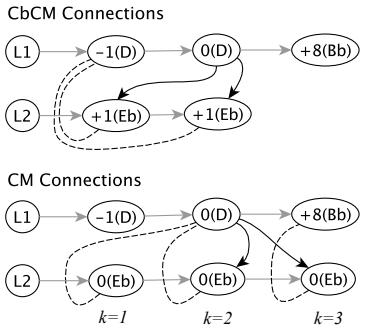

Figure 6. Different approaches to encoding form in the CbCM and the CM.

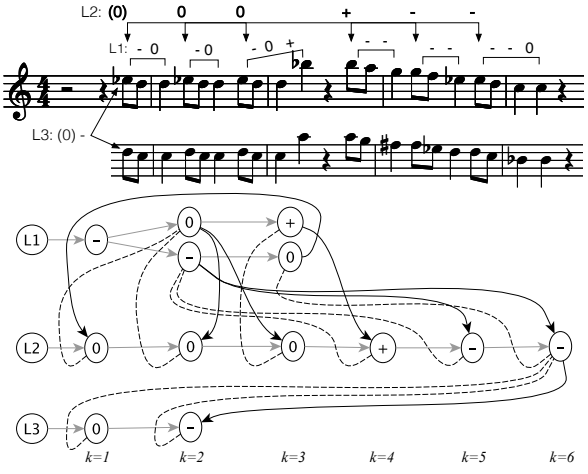

Figure 7. *Schema* view of a trained CM, showing *boundary links* (dotted) and *cueing links* (arrow).

5.4 The Production Module (PM)

The PM is responsible for generating musical output, with the underlying philosophy that music comprehension is supported by the retention of musical materials in WM. This idea can be related to Schmidhuber's theories of compression and creativity [24]. Through motivic exploitation—i.e., the use of musical parallelism—composers are able to guide listeners through the musical discourse, introducing thematic materials in an intelligible manner. Models that focus primarily on reproducing the transition probabilities of a training corpus generally fail to produce convincing results because they ignore musical parallelism (or achieve it only accidentally); the *IDyOM* model of Wiggins et al. is a case in point [25].

For our current purposes, we will discuss generation in terms of the *continuation* of a user-defined musical 'seed.' PM generation combines high-level planning from the CM with the integration of local contextual information from WM. There are two basic approaches that can be taken: 1) Generate a high-level plan and fill-in the surface details according to the plan (i.e., 'top-down'), and 2) Incrementally infer phrase-level plans, based on WM content, and gradually 'unfold' a high-level plan (i.e., 'bottom-up'). It has been suggested that compositional processes enlist both approaches, used in alternation [26].

The PM begins by running CM learning/inference on the user seed, so that the WM reflects whatever content and structure can be inferred from the seed. Generation is then treated as a *means-end reasoning* problem, where 'means' are represented by the state/contents of the WM and CM, and the 'end' is the generation of well-formed musical segments that reflect an appropriate degree of musical parallelism. Since parallelism is generally recognized in the context of higher-level form (see Figure 5.2.2), the PM must next determine a formal structure or 'plan.'

5.4.1 Formal Planning

In a 'top-down' approach to form generation, the PM first probabilistically generates a high-level path (or plan) $P_k^{L_n}$, using the transition probabilities encoded by the L_n edges. It then proceeds to 'unwind' a formal structure by following *boundary links* from L_n to terminal nodes $L_{n-1}\eta_k^i\theta$, and extracting paths of the form $(L_{n-1}\eta_1^i, \cdots, L_{n-1}\eta_k^i\theta)$. This approach can be used with a CM of arbitrary size.

In a 'bottom-up' approach, form generation begins from the *stream state* ${}^S\delta_t$ (i.e., determined by the contents of WM) then probabilistically unfolds the higher-level form. In this case, the probabilistic selection incorporates evidence from L_{n-1}, as provided by *cueing links* (described in Section 5.3). For example, if the user seed is the *segment* $\{\text{E}\flat\ \text{D}\ \text{D}\ \text{B}\flat\}$, and ${}^S\delta_t = \{L_1\eta_3^0(+), L_2\eta_1^0(0), L_3\eta_1^0(0)\}$ (in Figure 7) , we can see that the *cueing link* from $L_1\eta_3^0(+)$ provides support for a transition to $L_2\eta_4^0(+)$. Once we have chosen the L_2 transition, we can probabilistically generate the remainder of the L_2 path.

5.4.2 Motivic Exploitation and Parallelism

With a formal plan in place, the PM must next determine a degree of parallelism to guide the generation of L_1 segments. Looking carefully at Figure 7, we notice that the *link* structure encodes the parallelism of the melody. Here, both $L_2\eta_1^0(0)$ and $L_2\eta_2^0(0)$ have boundary *links* to $L_1\eta_2^0(0)$,

indicating that the same contour has been repeated. The parallelism of a given L_n path can thus be calculated as the ratio between the number of unique terminal nodes and the number of *boundary links* connecting those terminal nodes. However, it may be the case that two (or more) terminal nodes share intersecting paths, suggesting a subsegment relationship (i.e., $\{0\ -\ 0\}$ and $\{0\ -\ 0\ +\}$). In such cases, we calculate the fractional parallelism as the ratio between the subsegment and segment path lengths. This value is then used to scale the boundary count of the longer path's terminal node.

The calculated parallelism acts as a guide for the PM when deciding whether to re-use segments from WM, or to generate new segments from the CM. The PM regulates this decision dynamically, by comparing the parallelism of the current plan (an L_2 path) with the parallelism of the WM contents. If the plan's parallelism is greater than that of the WM, the PM will use WM segments during generation. However, if the WM's parallelism exceeds that of the plan, the PM will generate segments from the CM.

When using WM contents for generation, the PM makes a weighted probabilistic selection from the segments in WM, using cognitive salience as a weighting. Once a segment has been selected, its terminal node is used to extract a path from the CM. Note that this path will not necessarily produce a direct quotation, but it will replicate the **Schema** (i.e., contour) of the WM segment. The boundary event for the new segment will be derived from the current L_2 node in the formal plan, and the remainder of the segment will be derived from the L_1 path.

5.4.3 Feedback: Acting and Perceiving

In order to complete the agent metaphor, MusiCOG must not only *act* in its world, it must also perceive the results of its actions. To realize this, we feed the *segments* generated by the PM back into the PE. This is an important step, because it allows the model to evaluate its actions, and modify its behaviour accordingly, providing a form of *intentionality*. In the current version, this intentionality is limited to a very simple goal: to generate *segments* with well-defined boundaries, which support the PE's grouping mechanisms.

To achieve this, we include a routine to evaluate each transition in the generated segment using the stream separation/segmentation functions of the PE (see Section 5.1). If a given transition fails to produce the desired result—i.e., the beginning of a *segment* is not perceived as a boundary, or boundaries are percieved where they are not intended—the PM will attempt to find an alternative solution. When no solution can be found in the CM, a simple, rule-based generation routine attempts to create a transition that will satisfy the PE's requirements. This step has the added benefit of introducing novelty in a manner that is constrained by the overall goals of the system (i.e., PE grouping, *segment* structure, contour, and formal planning). Further, since novel transitions generated in this manner become inputs to the PE, they will be retained by WM and learned to some degree by the CM, leading to complex emergent behaviour.

6. IMPLEMENTATION DETAILS

In our implementation we handle rhythm using a separate CM level, placed at the bottom of the model (i.e., L_0). This level is trained on the rhythmic representations discussed in Section 4.2. The rhythmic information for boundary events (i.e., at L_2) is encoded by associating L_2 nodes with L_0 nodes, using a special *association link*. An association $\alpha(a,b)$ is a weighted connection between any two nodes a and b, the weight of which indicates how often both nodes have been visited at the same time step. During generation, L_0 paths are generated in the same manner as L_1 paths, and the boundaries are added using weighted probabilistic selection from the L_2 rhythmic associations. When the PM draws motivic material from WM, the rhythm is taken directly from the chosen WM segment.

In the current version, generation is monophonic. However, because CM learning/inference already build associations between *all* nodes visited at each time step, across *all* streams, experimentation with polyphonic generation will begin in the near future.

MusiCOG is the learning/generation system for our interactive composition software *ManuScore*. In the *ManuScore* interface, the user is given two options for controlling how MusiCOG will vary motivic material. With the default setting, the PM will draw all motivic material from WM, and will increases/decrease parallelism by choosing between high/low salience WM segments. When this option is switched off, the PM will reduce parallelism by generating novel segments directly from the CM.

7. PRELIMINARY FINDINGS

7.1 The PE, WM, and CM

We first trained MusiCOG on the Partita in A minor for solo flute, BWV 1013, by J.S. Bach. Each movement was trained separately, and the total number of inputs received, and nodes produced, was recorded as each movement was learned (see Figure 8). It will be noticed that, with the exception of the *Sarabande*, the percentage of nodes to inputs decreased with each movement, indicating the exploitation of previously learned material. Training on the *Allemande* added 1297 nodes for 3066 items of information (i.e., contour, interval, and pitch data for 1022 events), giving a total compression of 42%. Subsequent training on the *Courante* added 1006 nodes for 2673 inputs (compression $= 38\%$). The *Sarabande* added 427 nodes for 903 inputs (compression $= 47\%$) and finally, the *Bourée Anglaise*, added 431 nodes for 1587 inputs (compression $= 27\%$). Looking more carefully at the spike in node creation for the *Sarabande* it was noted that this work produced a significant number of new nodes during the *second* training pass. This is mostly likely due to changes in low-level boundary detection between the first and second passes (resulting from increased predictive success) leading to subsequent revision of the segment structure.

The chunking structure in WM produced by the opening of the *Allemande* of BWV 1013 is shown in Figure 9. Due to the influence of prediction on segmentation, hierarchical structure tended to improve with training. It was also

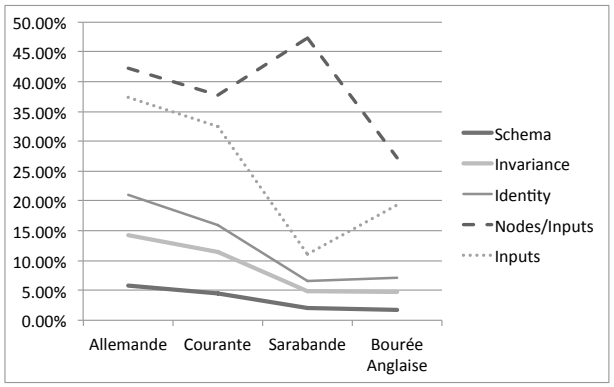

Figure 8. Nodes learned during training on each consecutive movement from Bach's BWV 1013. The "Inputs" line indicates the percentage of the total inputs contained in each movement. The top line indicates the ratio of nodes created to inputs (as a percentage), and the lower three lines indicate the nodes produced at each degree.

noted that, in some cases, L_1 chunks would fail to be chunked at L_2, resulting in an incomplete hierarchy. This could be solved by appending the isolated L_1 chunks to the preceding L_2 chunk.

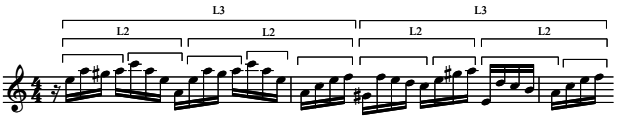

Figure 9. A sample of MusiCOG's hierarchical segmentation of the *Allemande* from Bach's BWV 1013.

To test stream separation we trained the system on the fugues from Bach's BWV 846 and BWV 853 from the Well-Tempered Klavier. Performance was generally acceptable, though the system did have difficulties with voice-crossings, which tended to occur at points where a single note was shared by two adjacent voices. Training on the 846 fugue alone added 1375 nodes for 2202 inputs, yielding a compression of %62, while subsequent training on the 853 fugue added only 1847 nodes for 4167 inputs (compression = %44), again indicating considerable expoitation of previously learned material.

Thorough testing of PE, WM, and CM performance will be the subject of a future study.

7.2 Generation in the PM

Screen capture videos of live generation within ManuScore can be viewed at the following url:

`http://dl.dropbox.com/u/8429426/index.html`.

All examples begin with a pre-composed musical 'seed', from which MusiCOG generates an extended continuation. Generation in all examples ends when MusiCOG's generated formal plan is completed. Because continuation was chosen as the compositional task, formal generation is carried out using the 'bottom-up' approach discussed in Section 5.4. Because the PE currently has no model of key or scale induction, it is not expected that generation will maintain appropriate key/scale relationships. Our interest rather is in the exploitation of motivic material, and the development of an overall formal structure.

In *Example 1*, trained on Bach's BWV 1013, we set the system to vary motivic material by generating novel segments from the CM. A pre-composed "seed" is given, which the system extends through continuation. The seed was written in the style of J.S. Bach, but was not directly drawn from the training material. It will be noticed that the system is quick to generate material not contained in the seed, but stylistically consistent with the training set. Of note also is the fact that the system is quick to exploit its own generated motivic material, and that, as the work develops, material is drawn from both the user seed and the system's own generated content.

In *Example 2*, MusiCOG is set to generate movitic variety by alternating between low and high salience motifs, resulting in more tightly intergrated motivic exploitation, and more limited motivic variety. It will be noted that, as generation is primarily interval and contour based, the music quickly departs from the key implied by the seed, as was expected.

In *Example 3* and *Example 4*, we trained the system on the solo part from Maxwell's conerto for flute and ensemble, *Vovere*. This is a single-movement work, for which the system created 3506 nodes for 4584 inputs, giving a rather modest compression of 77%. However, it was noted that the PE often produced exceptionally long L_1 segments, likely due to the fact that L_1 segmentation uses a fixed percentage of the standard deviation of *cohesion* as a boundary condition. A more dynamic boundary condition would maintain more appropriate segment lengths, which would likely improve compression.

The seed for these examples was composed by Maxwell, and does not quote directly from the training work. Motivic variety in both examples was provided by CM generation. Again, the continuation is stylistically consistent with the seed. *Example 4* offers a brief look at interactive composition with *ManuScore*. Because MusiCOG is an online cognitive agent, the user can begin playback at any point in the score and MusiCOG will infer from the played music and offer a continuation, in a manner similar to Pachet's *Continuator* [27].

We did notice that the use of parallelism tended to follow a somewhat predictable pattern of similarity and change, likely due to the use of a threshold-based mechanism for governing motivic exploitation. Determining parallalism incrementally, with each new L_2 node in the formal plan, would help break this pattern, potentially generating formal strategies more closely related to the training set.

8. FUTURE WORK

Of particular interest for the immediate future are the implementation of beat/tactus and key/scale induction in the PE, and polyphonic generation. We are also interested in further exploring the possibilities offered by PE evaluation of PM output. For example, PE evaluation could be con-

trolled parametrically, allowing generation to be 'steered' in real-time by the user.

We are also interested in the possibilities offered by introducing declarative knowledge (as metadata) into the learning process, in a manner similar to ACT-R. For example, the system could store instrumentation data during training, which could be used by the PM to help generate more idiomatic instrumental material. Finally, it would be valuable to give MusiCOG the capacity to learn from the user's decisions during interactive composition processes. For example, a form of reinforcement learning could be used to strengthen the probabilities of generated material retained by the user, or to suppress material rejected by the user.

9. CONCLUSION

We have presented an integrated cognitive architecture for music learning and generation called MusiCOG. Although other cognitively-grounded generative models exist [28,29], we believe MusiCOG to be the first integrated architecture, founded on ideas from the field of ICA design. We have provided some preliminary results indicating the performance of MusiCOG's constituent modules, and given samples of generation from a working implementation of MusiCOG in the *ManuScore* interactive composition environment.

Acknowledgments

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

10. REFERENCES

[1] H. Purwins, M. Grachten, P. Herrera, A. Hazan, R. Marxer, and X. Serra, "Computational models of music perception and cognition ii: Domain-specific music processing," *Physics of Life Reviews*, vol. 5, no. 3, pp. 169–182, 2008.

[2] H. Purwins, P. Herrera, M. Grachten, A. Hazan, R. Marxer, and X. Serra, "Computational models of music perception and cognition i: The perceptual and cognitive processing chain," *Physics of Life Reviews*, vol. 5, no. 3, pp. 151–168, 2008.

[3] H. Chong, A. Tan, and G. Ng, "Integrated cognitive architectures: a survey," *Artificial Intelligence Review*, vol. 28, no. 2, pp. 103–130, 2007.

[4] D. Vernon, G. Metta, and G. Sandini, "A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 2, pp. 151–180, 2007.

[5] J. Anderson, "Act: A simple theory of complex cognition," *American Psychologist*, vol. 51, pp. 355–365, 1996.

[6] R. Young and R. Lewis, "7 the soar cognitive architecture and human working memory," *Models of working memory: Mechanisms of active maintenance and executive control*, p. 224, 1999.

[7] P. Langley and D. Choi, "A unified cognitive architecture for physical agents," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 21, 2006, p. 1469.

[8] B. Snyder, *Music and Memory: An Introduction*. The MIT Press, 2000.

[9] N. Cowan, "What are the differences between long-term, short-term, and working memory?" *Progress in brain research*, vol. 169, pp. 323–338, 2008.

[10] A. Baddeley, "Short-term and working memory," *The Oxford handbook of memory*, pp. 77–92, 2000.

[11] P. Janata, B. Tillmann, and J. Bharucha, "Listening to polyphonic music recruits domain-general attention and working memory circuits," *Cognitive, Affective, & Behavioural Neuroscience*, vol. 2, no. 2, pp. 121–140, 2002.

[12] P. Langley, J. Laird, and S. Rogers, "Cognitive architectures: Research issues and challenges," *Cognitive Systems Research*, vol. 10, no. 2, pp. 141–160, 2009.

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

[14] J. Maxwell, P. Pasquier, and A. Eigenfeldt, "The Closure-based Cueing Model: Cognitively-inspired learning and generation of musical sequences," in *Proceedings of the 2011 Sound and Music Computing Conference*, 2011.

[15] E. Chew and X. Wu, "Separating voices in polyphonic music: A contig mapping approach," 2004.

[16] A. Jordanous, "Voice separation in polyphonic music: A data-driven approach," 2008.

[17] M. Pearce, D. Mllensiefen, and G. Wiggins, "Melodic grouping in music information retrieval: New methods and applications," *Advances in Music Information Retrieval*, pp. 364–388, 2010.

[18] I. Peretz and M. Babaī, "The role of contour and intervals in the recognition of melody parts: Evidence from cerebral asymmetries in musicians," *Neuropsychologia*, vol. 30, no. 3, pp. 277–292, 1992.

[19] E. Cambouropoulos, "Melodic cue abstraction, similarity, and category formation: A formal model," *Music Perception*, vol. 18, no. 3, pp. 347–370, 2001.

[20] C. Brower, "A cognitive theory of musical meaning," *Journal of Music Theory*, vol. 44, no. 2, pp. 323–379, 2000.

[21] I. Deliège, "Cue abstraction as a component of categorisation processes in music listening," *Psychology of Music*, vol. 24, no. 2, p. 131, 1996.

[22] ——, "Prototype effects in music listening: An empirical approach to the notion of imprint," *Music Perception*, vol. 18, no. 3, pp. 371–407, 2001.

[23] E. Cambouropoulos, "Musical parallelism and melodic segmentation," *Music Perception*, vol. 23, no. 3, pp. 249–268, 2006.

[24] J. Schmidhuber, "Driven by compression progress: A simple principle explains essential aspects of subjective beauty, novelty, surprise, interestingness, attention, curiosity, creativity, art, science, music, jokes," *Anticipatory Behavior in Adaptive Learning Systems*, pp. 48–76, 2009.

[25] G. Wiggins, M. Pearce, and D. Müllensiefen, *Computational Modelling of Music Cognition and Musical Creativity*. Oxford, UK: Oxford University Press, 2009.

[26] D. Collins, "A synthesis process model of creative thinking in music composition," *Psychology of Music*, vol. 33, no. 2, p. 193, 2005.

[27] F. Pachet, "The continuator: Musical interaction with style," *Journal of New Music Research*, vol. 32, no. 3, pp. 333–341, 2003.

[28] G. Wiggins, "'computer models of musical creativity'david cope," *Literary and Linguistic Computing*, 2007.

[29] G. Assayag, G. Bloch, M. Chemillier, A. Cont, and S. Dubnov, "Omax brothers: a dynamic topology of agents for improvization learning," 2006.