BeatBender: Subsumption Architecture for Autonomous Rhythm Generation

Aaron Levisohn
Simon Fraser University
250-13450 102 Ave.
Surrey, British Columbia V3T 0A3
+1 604 782 7474
alevisoh@sfu.ca

Philippe Pasquier
Simon Fraser University
250-13450 102 Ave.
Surrey, British Columbia V3T 0A3
+1 604 782 7474
pasquier@sfu.ca

ABSTRACT
BeatBender is a computer music project that explores a new method for generating emergent rhythmic drum patterns using the subsumption architecture. Rather than explicitly coding symbolic intelligence into the system using procedural algorithms, *BeatBender* uses a behavior-based model to elicit emergent rhythmic output from six autonomous agents. From an artistic perspective, the rules used to define the agent behavior provide a simple but original composition language. This language allows the composer to express simple and meaningful constraints that direct the behavior of the agent-percussionists. From these simple rules emerge unexpected behavioral interactions that direct the formation of complex rhythmic output. What is striking is that these rhythmic patterns, whose complexity is beyond human grasp, are both musically interesting and aesthetically pleasing. The output from the system is evaluated using both subjective and objective criteria to assess degrees of complexity, convergence, and aesthetic interest.

Categories and Subject Descriptors
H.5.5 Sound and Music Computing: Systems

General Terms
Performance, Experimentation, Aesthetics

Keywords
Sound and Music, Rhythm, Generative Art, Metacreation, Subsumption Architecture, Aesthetics

1. INTRODUCTION
The development of artificial intelligence (AI) techniques have allowed artist to develop computational works that are capable of autonomous creative behavior. Such artworks are now commonly referred to as metacreations [1]. Unlike traditional artistic methods in which the artist maintains control over the final form and representation of their work, these new methods are more open ended. The artist's role in development of a metacreation becomes that of the programmer, developing a system that will ultimately produce the artwork itself.

The ability of computers to exhibit intelligent and creative behaviors has been demonstrated through their use in solving problems beyond human capabilities; they have produced paintings [2] and music [3]; and they have outwitted the best human chess players [4]. While questions remain about the nature of intelligence and how machine intelligence differs from biological intelligence, it is now an accepted fact that computers can exhibit behavior that would be considered intelligent if demonstrated by a human. The questions now being asked seek to define the limits and possibilities of computational intelligence.

Working with metacreations allows for the application of AI techniques in unusual and interesting ways. By repurposing AI techniques, developers of metacreations produce works that are highly engaging and aesthetically pleasing. These explorations also provide new methods for applying AI techniques that can be utilized in scientific, economic, and entertainment applications.

Metacreations have utilized various AI techniques in accomplishing specific goals. Often, multiple AI techniques are used collectively to engender greater complexity or creative behavior. Artificial neural networks, autonomous agents, cellular automata (CA), genetic algorithms/programming, as well as various artificial life techniques have all been utilized in the production of metacreations.

This paper presents a computer music project called *BeatBender* that explores a new method for generating rhythmic drum patterns using the subsumption architecture developed by Rodney Brooks [5]. Rather than explicitly coding symbolic intelligence into the system using procedural algorithms, *BeatBender* uses a behavior-based model that facilitates emergent expression. Subsumption architecture systems are designed to respond to stimuli in the environment using a hierarchical set of rules. Depending on the state of the environment, different rules of varying complexity are invoked, generating the behavioral output of the system. From an artistic perspective, the rules used to define the agent behavior provide a simple but original composition language. This language allows the composer to express simple and meaningful constraints that direct the behavior of the agent-percussionists. From these simple rules emerge unexpected behavioral interactions that

direct the formation of complex rhythmic output. What is striking is that these rhythmic patterns, whose complexity is beyond human grasp, are both musically interesting and aesthetically pleasing.

The *BeatBender* system is comprised of six autonomous agents, each responsible for the production of one beat of every measure in an ongoing rhythm. Each agent is designed to react individually to a set of perceptions at different, but sequential, points in time. The interaction between agents is determined by rules which are encoded into the layers of the subsumption architecture. The agents' behaviors, which are determined by their collective interaction, generate audio output in the form of complex and aesthetically interesting rhythmic patterns. This paper describes the development of the *BeatBender* system and the explorations undertaken using the subsumption architecture to design a multi-agent system capable of eliciting emergent behavior.

2. BACKGROUND

While rhythm generation is not a new subject in computer music, the use of the subsumption architecture as the foundation for such a system has never been tried. The subsumption architecture is well suited for rhythm generation since the articulation of beats does not require any form of symbolic representation. The intuitive nature of rhythm production has been demonstrated by researchers studying the evolutionary basis of human and cross-species rhythmic abilities [6]. Using the subsumption architecture allows autonomous agents in a closed system to exhibit emergent behavior through their collective interaction. By prioritizing the interaction among behaviors, the subsumption architecture is an ideal tool to model the ways in which participants in a drum circle react and adapt to each others' distinct playing styles. This results in a process of rhythm development that is both emergent and dynamic.

2.1 Subsumption Architecture

An agent is a computer system or component capable of independent autonomous action [7]. Agents are self-directed and perform actions within an environment in order to meet their design objectives. A multi-agent system consists of a number of agents that are capable of interacting to meet an individual or collective goal. *BeatBender* is a multi-agent system in which agents respond to each other's behaviors in order to generate rhythmic output. Rather than explicitly coding each agent's behavior ahead of time, *BeatBender* constrains the agents' actions through the use of the subsumption architecture.

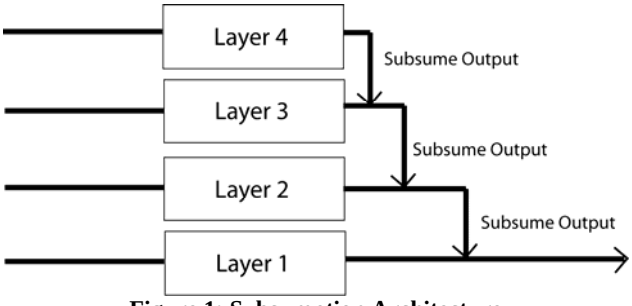

Figure 1: Subsumption Architecture

The use of the subsumption architecture as a means of eliciting emergent behavior in robots was introduced by Rodney Brooks in his seminal paper "Intelligence without Representation" [5] Rather than utilizing explicitly programmed algorithmic solutions, subsumption architecture systems focus on developing sets of behaviors; these behaviors are implemented as layers, with some layers given priority over others. Low level layers are given the lowest priority and usually invoke non-critical behaviors. Higher level layers generally invoke behaviors that are more complex and often vital to the functioning of the system (Figure 1). Specific layer behaviors are often selected based on environmental data gathered using sensors; however other data can be used as well.

While the subsumption architecture methodology has been primarily utilized in the field of robotics it is equally applicable to other endeavors in which emergent system behaviors are desired. For example, Bryson et al used the architectures in the development of a musical accompanist [8] and Nakashima and Noda incorporated it into their design of intelligent agents capable of playing soccer games [9]. While the subsumption architecture has been applied in musical systems, it has been solely toward the production of tonal output. *BeatBender*, uses the subsumption architecture for the purpose of exploring emergent rhythm generation.

2.2 Prior Work

As a metacreation, *BeatBender* positions itself within two different disciplines. First, *BeatBender* must be situated among works that explore rhythm generation. Second, it must be situated among general electronic music applications that utilize AI techniques.

Rhythm generation applications take various forms and serve a variety of purposes. In general, most of these types of applications are designed to produce rhythms as accompaniment to a human performance. One example of this is *Haile*, the anthropomorphic robotic percussionist designed by Weinberg and Driscoll [10]. *Haile* is capable of both mimicking and collaborating with a live human performer. One of Weinberg's primary goals was the implementation of a system that would allow Haile to develop meaningful representations of music in real-time. *Haile* uses complex analysis software to both establish the characteristics of the live performance and to compute rhythmic output. One of the benefits of subsumption architecture is the significant reduction of computational load since there is no need for symbolic representation. Complex behavior can be elicited using simple behavior-based rules.

Several systems, including Eigenfeldt's Kinetic-Engine [11], utilize agents as a means of representing individual drummers within a composition. Like *BeatBender*, *Kinetic-Engine* is a metacreation and requires no real-time input from a user. *Kinetic-Engine* uses networked agent architecture to emulate a percussive ensemble. When activated, the system assigns "personalities" to the agents who collectively personify the human elements in a drum circle. Eigenfeldt's agent system requires a much higher level of complexity than systems utilizing the subsumption architecture. *Kinetic Engine* uses a social model of agent interaction to emulate the behavior of live human performers. Agents "make eye contact" with one another

and, once connected, adjust to each other's performance. In addition, a special agent type called a conductor is used to oversee high level organizational elements. The need for a centralized agent model demonstrates the complexity of the system. Eigenfeldt claims that this level of complexity is necessary in order for a system to be "musically successful." He cites Brown's work [12] with rhythm generating CA as evidence of this. BeatBender explores emergence as an alternate means of generating musically successful rhythmic output using a decentralized agent model.

Pachet also developed a multi-agent system for generating and evolving rhythms [13]. Pachet's system has several similarities to *BeatBender* including its use of a rule-based approach to rhythm generation. However, unlike *BeatBender's* rules which are layered within the subsumption architecture, Pachet's rules explicitly direct patterns towards a particular type of known structure (e.g. a rock beat). While Pachet's rules provide a greater degree of control and allow for the shaping of the rhythms in much more direct ways, they do not provide an opportunity for emergent behaviors to develop.

The use of multi-agents systems is only one of many AI techniques that can be used in rhythm exploration. As noted earlier, Brown explored the rhythm generating potential of cellular automata (CA) [12]. One of the properties of CA is their tendency to exhibit predictable types of behavior. In Brown's paper he describes various rules and techniques that he uncovered during his exploration of linear CAs. His techniques are categorized based on the resultant behavior. Some behaviors include: rhythmic inversion, density thinning, evolving inversion, and emergent cycles. Brown concludes that these techniques result in patterns that lack musical meaning, and are more "intellectually fascinating" than aesthetically valuable. While *BeatBender* does not use CA explicitly, the linear relationship between agents results in similar behavior. However, there are two major differences between *these types of systems;* First, CA cells are processed in parallel while *BeatBender's* agents are processed in series. Second, *BeatBender's* subsumption rules implicitly encode system intelligence, allowing for the emergence of adaptive and contextual behavior. CA rules, on the other hand, are hardcoded prior to system activation, inhibiting responsive behavior.

Miranda [14] also used CA as the basis for a musical composition system. Rather than focusing on rhythms, however, he produced two systems capable of producing complete instrumental pieces. CAMUS generates MIDI notes using 2 discrete CA models: The Game of Life and Demon Cyclic Space. These two CAs work together to generate notes and arrange them within a composition. *Choasynth* uses CA to control a granular synthesis process based on the way electrical current flows through capacitors. Each cell controls the frequency of an oscillator that determines the makeup of a particular sound. The complexity of Miranda's systems is necessary to overcome the predictable nature of CAs and to allow for the development of rich musical output. Much of this complexity, however, is required solely to add structure to compositions and not in the process of musical generation. Additionally, the techniques used in this process, such as bitwise operations, utilize arbitrary mappings that do not rely on computational intelligence. The subsumption architecture alleviates the need for such arbitrary mappings by encoding intelligence directly into the agents.

Other techniques including neural networks and genetic programming/algorithms have also been applied to the production of computationally generated rhythmic patterns. Dolson describes the use of a neural network to both classify and generate rhythmic patterns [15]. Neural network systems, however, are not capable of emergent behavior since they are only able to replicate and combine rhythms that they have previously been trained to identify.

Tokui and Iba demonstrate the use of interactive evolutionary computation (IEC) as a means of producing rhythms in their CONGA system [16]. This system demonstrates the complexity that is often required to design a system that produces musically successful rhythmic output. The CONGA system uses a combination of both genetic programming and genetic algorithms to produce rhythms. *BeatBender* aims to accomplish similar goals using the subsumption architecture which alleviates the need for such a high degree of complexity.

3. SYSTEM DESCRIPTION

BeatBender is an autonomous rhythmic sequencer utilizing a scalable multi-agent system. Scalability is desirable property of any distributed system, as it is a good indicator of its decentralized nature. In this paper, we limit ourselves to six agents but this number can easily be adapted for different needs. Each agent in the system is responsible for the generation of one beat of every measure of continuous rhythmic output. At any given time the agents can be in one of two states: ON or OFF. During initialization all agents are set to OFF. When the system is started, each agent is activated one at a time in sequence. An agent's state is determined by a set of conditional rules that are applied at the exact moment of activation. These rules are implemented using the subsumption architecture (presented in section 2.1). If, after processing, the agent's state is set to ON, a drum sound is triggered; if the agent's state is set to OFF no sound is produced.

3.1 Design Considerations

Using the subsumption architecture for the *BeatBender* project requires two distinct design phases: building the architecture, and implementing the specific layers used by the subsumption system. The initial phase involves the implementation of a robust tool that allows for the exploration of various possible layer rules. These rules become the basis for the behaviors that each agent is capable of performing. This tool incorporates a flexible interface for enabling and disabling behaviors as well as adjusting layer hierarchies. Additionally, it includes a graphical representation as well as a text output system to record the specific output of the generated rhythmic patterns for analysis.

The second design phase utilizes the tool described above to explore subsumption rules and layer configurations in order to elicit emergent behavior. These behaviors were evaluated based on the characteristics of the rhythmic output.

3.2 The BeatBender Agents

The *Beatbender* system consists of six independent agents, each one capable of triggering audio events. The agents have states which can be set to either ON or OFF. When the system is activated, the state of each agent is computed in sequence. As each agent is activated, the subsumption architecture determines which behaviors to perform based on the current configuration of the system and sets the agent's state to be either on or off. At any time multiple behaviors can be triggered, but only the one on the highest subsumption layer is enacted to determine the agent's new sate. Once the new state has been set the system updates the environment variables which include the state of each agent as well as the total number of agents in the ON state. If a particular agent's state is set to ON then an audio event is triggered and one of three drum sounds plays as determined by the subsumption layer rules[1].

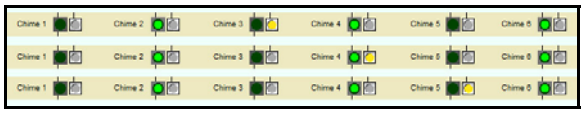

Figure 2: The state of all six agents over three time steps.

The current activity of the system is visually depicted using a display on the main interface (Figure 2). The display depicts all six agents in a horizontal row. A yellow dot next to each agent representation indicates when a particular agent is being activated. The green dot next to each agent representation indicates the current state of the agent. If the green dot is present, then the agent is in its ON state indicating that it triggered a drum sound on the last cycle. Figure 3 shows the portion of the interface that displays each agent's respective state. In the illustration, the interface has been replicated three times in order to show three sequential time steps.

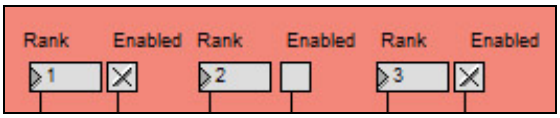

Figure 3: The Rules Interface

3.3 Subsumption Layer Tools

As well as providing a visual representation of the state of each agent in the system, the *BeatBender* software also includes an interface to assist in the exploration of the system behavior (Figure 3). This interface allows specific layers to be turned on or off and for the rankings of individual layers to be adjusted. Layers that are assigned lower ranks are subsumed by layers with higher ranks. This forms the basis for the subsumption architecture. Figure 3 shows the setting for three layers. In the illustration, layers 1 and 3 are both activated. The interface allows for up to 10 layers to be explored simultaneously.

3.4 Analysis Tools

The *BeatBender* interface also consists of a graphical representation of the last six patterns played by the system. This permits quick analysis of the rhythmic structure that the current system configuration is generating (Figure 4). Each horizontal row illustrates one cycle of the system with the state of each agent indicated by a red dot. The vertical rows illustrate each particular agent's state over time. The first full cycle is displayed in the top row with each subsequent cycle appearing one level lower. After all six rows are filled the display return to the top row.

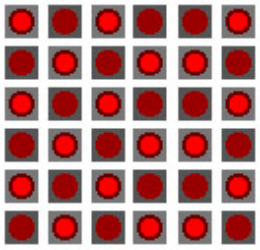

Figure 4: Representation of current beat pattern

The interface also captures each rhythm as a text file. This allows for more complex rhythmic structures to be identified and analyzed. Rhythms are coded using an "X" for every ON beat and a "_" for every rest (OFF beat). Using this scheme the rhythm illustrated in Figure 5 would be output as:

```
X _ X _ _ X
_ X _ X X _
X _ X _ _ X
_ X _ X X _
X _ X _ _ X
_ X _ X X _
```

Both the layer tools and the analysis tools are particularly important for working with a system utilizing the subsumption architecture. In order for the system to produce emergent rhythmic patterns, numerous configurations of behaviors and rankings need to be tried. Each layer of the architecture must be implemented, tested, and modified in order to identify configurations that produce complex agent interactions, and consequently, interesting rhythmic output. Typically, in subsumption architecture systems, the layers are implemented linearly, starting from the zeroth layer (the lowest level) and ending with the highest level layer. For *BeatBender*, though, the layer tool allows for the exploration of the various permutations by providing a method for quickly reordering layer rules. In a system such as this where the success of the behaviors is based partially on aesthetic criteria, this interface was an essential element.

The analysis tools were also essential for evaluating the output produced by the system. While the success of the *BeatBender* system was based partially on subjective criteria, the objective analysis of the rhythms was accomplished by looking for patterns within the rhythmic structures. The graphical representation of the rhythms and, in particular, the logging of each rhythm as text, enabled this quantitative analysis to be done.

[1] While only 3 sounds were used for this implementation of *BeatBender*, there is no limit to the number of drum sounds that can be used.

4. SYSTEM COMPONENTS

4.1 The Agents

A single *Agent* was developed to represent each of the drummers in the system. This agent was instantiated six times to create the full *BeatBender* model.

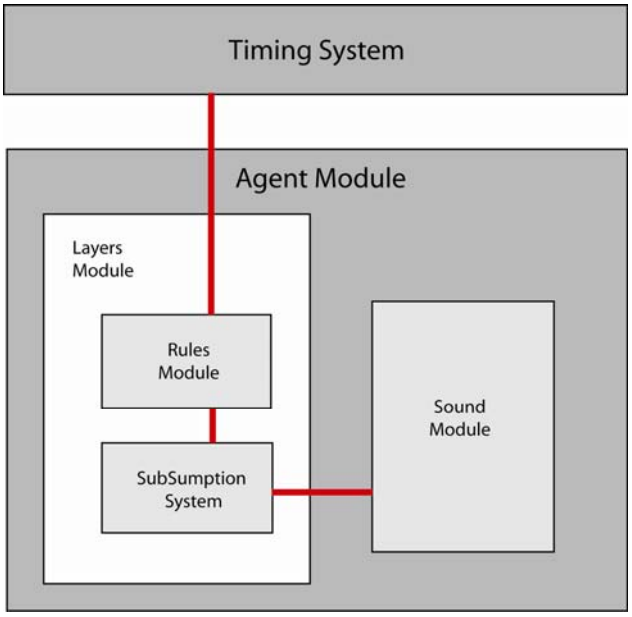

Figure 5: System Diagram

The *Agent* module acts as a container for the other components that are required for the system to function. This includes the *Layers* module and the *Sound* module. The *Layers* module implements the entire subsumption architecture system and the *Sound* module implements the audio playback system (Figure 5). Only the system responsible for controlling meter and timing was implemented separately from the *Agent* Module. This *Timing system* is responsible for the sequential activation of each agent and controlling the rate at which the agents are triggered.

4.2 System Goals

While applications that use the subsumption architecture to control robots have specific system goals such as avoiding objects, the goals for the generation of interesting rhythmic patterns are less straightforward. The subjective nature of assessing the quality of rhythmic patterns also presents additional difficulties. Despite these challenges, specific criteria were established to both subjectively and objectively assess the quality of the rhythms generated by the system.

For *BeatBender*, the goal state is one in which the agents exhibit emergent behaviors and the resulting rhythmic output displays recognizable repeating rhythmic patterns. The rhythms produced in such a scenario should not be radically different from one time step to the next, but rather should slowly evolve while maintaining some regularity. This development of rhythmic patterns with these characteristics will be dependent upon the rules implemented within the layers of the subsumption architecture.

The rhythmic output from *BeatBender* is also evaluated on aesthetic criteria. This assessment takes into account both the quality of the rhythmic pattern produced by the system and the instrumentation used to play it. The quality criterion assesses the musicality of each rhythm as well as its perceived complexity. This criterion assesses the aesthetics of the rhythm's structure regardless of how it is performed. The instrumentation criterion evaluates the musicality of the particular drum sounds used to play the rhythm. While the instrumentation does not affect the beat pattern produced by the system, it does alter the listener's perception of the rhythmic structure. Successful instrumentation will use the available drum sounds to produce the effect of multiple drummers performing together. Rather than sounding like a single rhythm being performed, the listener should perceive several discrete rhythmic patterns, reflecting the responsive nature of the agents within the system.

4.3 Agent Behaviors

Agents can perform only two types of behaviors: they can change their state and they can select a drum sound to play. When an agent implements a state change it can turn ON, turn OFF, or FLIP its state. If, after processing, an agent is set to the ON state then it also must select a drum sound to use for playback. The current implementation of *BeatBender* provides three sound options, each one a different variation of a tabla drum: a long hit, a solid hit, and a low hit. These sounds were specifically selected and sampled from live instruments in order to produce a more natural rhythmic output.

4.4 Agent Perceptions

Each agent is continually updated with specific perceptions which are used to determine which behaviors are enacted under which conditions. Rather than utilizing the physical sensors that a robot would use, *BeatBender* maintains a representation of the current virtual "environment" using a set of perceptions that are updated when an agent is activated. These perceptions include: the agent's state, the previous agent's state, the next agent's state and the number of currently active agents in the system. Each one of these perceptions is used to determine which behavior should be enacted by the system by running them through a series of conditional statements. These conditional statements and the behaviors they enact are the basis for the subsumption layers that determine agent behavior. Using agent perceptions in this way replaces the need for the physical sensors that are used in robotic subsumption systems.

4.5 Layers and Rules

Layers are made up of one or more rules. Each rule has an antecedent and a consequence. The antecedent provides the precondition necessary for a rule to be instantiated. Antecedents use agent perceptions to test if specific preconditions have been met. The consequence selects the specific behaviors to enact if a precondition is met.

Each layer is also given a rank value. The rank value is used to set the order in which rules are processed. Rules on layers with higher rank values will supersede lower level rules. If the preconditions for two rules are met simultaneously, only the rule associated with the highest ranked layer will have its behaviors enacted.

Rule antecedents can be of two kinds: general and specific. Specific antecedents are designed to trigger behaviors only under very specific conditions. Rules with these types of antecedents tend to be given high rank values to ensure that they are activated when a specific condition arises. Rules with general antecedents, on the other hand, are usually given low rank values since they will be triggered under numerous conditions. Ranking rules this way ensures that these rules behaviors will only be instantiated if all the other rules have been passed over. A balance between general and specific antecedents is necessary for the *BeatBender* system to take advantage of the subsumption architecture and to develop interesting rhythmic patterns.

The rules used in *BeatBender* can be divided into four categories: *Collective*, *Directed*, *Temporal* and *Undirected*.

Collective rules use information about the total number of active agents in the system. An example of this type of rule would be: *If there are more than 3 active agents then turn this agent on.* In order to make transcription of rules simpler this same rule can be written in shorthand like this:

IF TOTALAGENTS > 3 THEN ON

Directed rules are based on information about an agent's specific neighbor or neighbors. An example of such a rule would be: *If the previous agent is on then flip this agent's state.* This could be shorthanded as:

IF P = ON THEN FLIP

Temporal rules are based on information about a agent's state over time. These rules can either track the number of *consecutive* ON and OFF states an agent has been in, or alternately, the total number of times an agent has been in either the ON or OFF state. An example of this type of rule is: *If the agent has been on for the last four cycles then turn it off.* This can be shorthanded to:

IF CONSECUTIVE > 4 THEN OFF

Undirected rules are based on information about an agent and its neighbors but without reference to a specific agent's state.
These rules are based on the techniques described by Brown in his paper *Exploring Rhythmic Automata* [12] which are themselves based off of Stephen Wolfram`s classification of CA behavior in general. These types of rules take into account the states of the current agent, previous agent and next agent (P, C, and N) the sum of which is used to assess local activity. If an agent is on then its value is 1; if it is off its value is 0. Unlike other types of rules, *Undirected* rules often include multiple conditions for each possible outcome (0 – 3) with each triggering a different behavior. The possible behaviors in this case are: turn on (ON), turn off (OFF), leave unchanged (U) or ignore (I). It is important to note the significant difference between leaving an agent unchanged and ignoring it. This difference illustrates the functionality of the subsumption architecture. When a condition statement results in the *Ignore* behavior, no action is taken. This allows rules on lower layers to get processed and to enact alternate behaviors. When a condition statement results in the *Unchanged* behavior, however, an agent's state is set and lower level rules do not get processed. An example of an *Undirected* rule is:

0 → ON, 1 → U, 2 → I, 3 → OFF

In this example, if the sum of the three agents' states is 0 then the agent is turned on. If the sum is 1 the agent's state is unchanged. If the sum is 2 the agent is ignored. If the sum is 3 the agent is turned off.

5. EVALUATION

BeatBender was evaluated on two distinct criteria: *emergence* and *aesthetics*. The emergence criterion was assessed using simple objective measures; the aesthetic criterion was assessed using subjective measures.

5.1 Emergence

The objective evaluation of *BeatBender* was done to assess the system's ability to exhibit interesting emergent behavior. Since *BeatBender* is comprised solely of agents capable of independent action, emergence manifests through the interaction between agent's behaviors. The emergent characteristics of the multi-agent system's behavior are revealed in the rhythmic pattern generated by the system as it converges towards equilibrium. These characteristics exhibit themselves in the form of convergent and recurrent structures within the generated rhythmic patterns. A comparison of different rule configurations can be done by evaluating the *quality of convergence* based on two factors: the length of the *converging* pattern and the length of the *recurring* pattern.

An analysis of the patterns produced by *BeatBender* was completed using the visual representation of the rhythmic output generated with the built-in assessment tools described in section 3.5. Rules were evaluated individually first, then in combination.

5.1.1 Individual Rules

The application of individual rules does not produce interesting emergent behavior since isolated rules immediately set the system into a state of equilibrium. Only though the interaction of two or more local elements is the process of convergence made visible. Individual rules do, however, produce rhythms that reveal two distinct types of behavior: *Consistent* and *Cyclical*. The multi-agent system is said to exhibit *Consistent* behavior if each agent maintains the same state in every cycle. The multi-agent system exhibits *Cyclical* behavior if the agents' states alternate on two or more subsequent cycles. The following chart shows rules that produce both types of behaviors:

Rule Applied	Pattern Generated	Behavior
IF P=OFF then ON	X _ X _ X _	Consistent
IF N = OFF then ON	X X X X X _	Consistent

IF P=N then FLIP	X _ X _ X X X X X X _ X _ X _ _ _ _ _ _ X _ X _ X X _ _ _ _ X X _ X _ _ X X X X _ _	Cyclical
IF P = OFF then FLIP	X _ X _ X _ _ X X _ _ X _ _ _ X _ _ X _ X X _ X X _ _ _ X X X _ X _ _ _ _ X X _ X _ X X X _ _ X X X X _ X X X X X _ _ _ _ _ _	Cyclical

5.1.2 Rule Combinations

Using rules in combination with each other facilitates emergent agent behaviors and the resulting complex rhythmic output. While many combinations do result in complex behaviours, others produce rhythms similar to those generated by individual rules. The following chart shows rule combination that resulted in the production of simple cycling or recurring patterns. In these examples the rule numbers corresponds to subsumption layer ranks.

Rules Applied	Pattern Generated	Behavior
#1: IF P = OFF then ON #2: IF N = OFF then ON	X X X X X _	Consistent
#1: IF P = OFF then FLIP #2: IF N = OFF then ON	X X X _ _ _ X X X X X _ _ _ _ _ X X X X X _	Cyclical

Many rule combinations do result in the expression of emergent behavior by the agents. In such cases, the complex interactions between agents results in a period of transition as the pattern converges towards equilibrium. During this phase the rhythmic output from the system converges towards a repeating pattern. The rhythmic patterns generated by these rules are classified based on the length of an initial converging pattern and the length of subsequent cycling pattern as depicted here:

Rules Applied	Pattern Type	Pattern Generated
#1: IF P = OFF then Flip	Converging (36 Beats)	_ X _ X X X X X X X _ _ _ _ _ _ _ X _ X _ X _ _ X X _ _ X _ _ _ X X X _
#2: IF P=N then FLIP	Cycling (36 Beats)	X _ _ X X X X _ X _ _ X X X X _ X _ _ X X X X _ X _ _ _ _ _ _ _ X _ X _

When multiple rules are combined together, the rhythms generated by the *BeatBender* system become increasingly complex. This results in longer converging patterns as well as longer cycling patterns. The cycling period for rhythms often becomes so long that the repetitions are indiscernible to human ears and patterns appear to transform continuously. Only through textual analysis are patterns discernable. An example of a pattern exhibiting this type of emergent behavior is generated using the following set of rules:

Rule 1: TURN ON AGENT
Rule 2: IF P = N then FLIP
Rule 3: 0→IGNORE, 1→ON, 2→IGNORE, 3 →REST
Rule 4: IF STRIKES > 4 then OFF
Rule 5: IF ACTIVEAGENTS > 3 then FLIP

The converging pattern produced by the interaction of these rules has a length of 8,535 beats. The cycling pattern has a length of 29,852 beats. (An audio sample of this rhythm can be heard at www.aaronlevisohn.com/beatbender.)

5.2 Aesthetics

The aesthetic assessment of *BeatBender* is based on the perceived musicality of each rhythm in terms of both pattern complexity and instrumentation. The process of evaluation is iterative and involves comparisons between subsequently generated patterns. The evaluation of a rhythm's complexity is accomplished by listening for motifs within the pattern. While all of the rhythms do eventually converge to cycling patterns, the more interesting ones take time to reach equilibrium. During the convergent period of such rhythmic production, subtle *shifts* and *transformations* can be identified. The *shifts* produce offsets of recurring patterns in a manner that allows musical phrases to repeat in non-predictable ways. Many rhythms repeat musical phrases with only slight modifications such as the flipping of a single beat from ON to OFF or vice-a-versa. These *transformations* result in slowly evolving patterns that exhibit variability without collapsing to chaos. (An audio recording and visual example of a pattern with a motif exhibiting *shifts* and *transformations* can be seen at www.aaronlevisohn.com/beatbender.)

In addition to identifying the complexity of the patterns produced by *BeatBender*, rhythms are also assessed for the aesthetic quality of their instrumentation. As described in section 4.3, drum sounds are set using particular rules within the subsumption layers. Having a separate set of rules for this purpose allows for the exploration of instrumentation to be undertaken separately from the rhythm generation itself. Exploring the aesthetic possibilities of a particular rhythm is accomplished by adjusting the *sound identifier* rules. These rules, which result from the interaction between all the agents, become increasingly complex as additional layer rules are

applied. Due to this, the drum sound associated with a specific agent cannot be set, but alternate system configurations can be explored. Successful configurations result in a combination of drum sounds that appears to be generated by multiple drummers performing together. The instrumentation used for the final implementation of *BeatBender* was discovered by exploring numerous variations of the *sound identifier* rules in order to achieve musical output that best met these aesthetic criteria.

6. CONCLUSION

Computers have been used for decades as tools to assist in the creation of art. As new computational techniques are developed, artists are among the first to explore them. This often results in new and innovative applications of these techniques. As AI techniques have become more accessible, they too have been applied to the production of artworks. These new artworks demonstrate a conceptual shift in the way that artists envision their relationship with computers. Using AI techniques, it is now possible to create computational works that are capable of autonomous creative behavior, blurring the line between artist and tool. Works of this sort explore machine creativity and intelligence while simultaneously addressing issues relating to human consciousness. With both aesthetic and research purposes, these projects bridge the gap between art and science. Such artworks are called metacreations.

This paper presents the development of a metacreation that uses the subsumption architecture to elicit emergent behavior. The work is comprised of six identical agents capable of enacting specific behaviors to change the overall state of the system. Despite the overall simplicity of the system, the interaction of these agents produces complex emergent behaviors that are expressed as rhythmic output from the system. The ability of the system to produce patterns of extreme complexity using a simple implementation model presents numerous possibilities for the exploration of machine creativity.

While this paper presents the results of a successful initial evaluation of the *BeatBender* system, a more robust evaluation is planned for the future. Since drumming has its origins in human song and dance rituals, this next evaluation will assess *BeatBender's* rhythmic output within a more natural human context. This evaluation will compare the rhythms generated by *BeatBender* to those produced by human performers. The results of this evaluation will help direct future work on the *BeatBender* project.

This implementation of *BeatBender* demonstrates a new method for generating emergent rhythms using the subsumption architecture. The subsumption architecture provides an easy yet powerful method for directing the composition of rhythms using simple sets of rules. From an artistic perspective, these rules function as a compositional language that permits the user to express meaningful constraints that result in unexpectedly complex and aesthetically pleasing rhythmic patterns. Future implementations of *BeatBender* will expand the layer rules to develop behaviors that encode specific musical structures. This will be a preliminary step in a larger move to make the system fully interactive and ultimately capable of real-time collaboration with live performers.

To listen to examples of the rhythms produced by the *BeatBender* system and to see visual representations of additional rhythmic patterns please visit www.aaronlevisohn.com/beatbender.

7. REFERENCES

[1] M. Whitelaw, *Metacreation: Art and Artificial Life*, The MIT Press, 2006.

[2] H. Cohen, "The further exploits of Aaron, painter," *Stanford Hum. Rev.*, vol. 4, 1995, pp. 141-158.

[3] D. Cope, "Computer modeling of musical intelligence in EMI," *Computer Music Journal*, vol. 16, 1992, pp. 69-83.

[4] D. Levy and M. Newborn, *How computers play chess*, Computer Science Press, Inc., 1991;

[5] Brooks, R.A., "Intelligence Without Representation," *Artificial Intelligence Journal*, vol. 47, 1991, pp. 139-159.

[6] J. Bispham, "Rhythm in Music: What is it? Who has It? And Why?," *Music Perception*, vol. 24, Dec. 2006, pp. 125-134.

[7] Woolridge, M. and Jennings, N.R., " Intelligent Agents: Theory and Practice," *Knowledge Engineering Review*, vol. 10(2), 1995.

[8] J. Bryson, A. Smaill, and G.A. Wiggins, "The Reactive Accompanist: Applying Subsumption Architecture to Software Design," Research Paper 606, Dept. of Artificial Intelligence, Edinburgh, 1992.

[9] H. Nakashima and I. Noda, "Dynamic subsumption architecture for programming intelligent agents ," *Multi Agent Systems*, 1998, pp. 190-197.

[10] G. Weinberg and S. Driscoll, "Robot-human interaction with an anthropomorphic percussionist," *Proc.of SIGCHI conf. on Human Factors in computing systems*, Montréal, Québec, Canada: ACM, 2006, pp. 1229-1232;

[11] A. Eigenfeldt, "The Creation of Evolutionary Rhythms within a Multi-agent Networked Drum Ensemble," *Proc. Intern. Comp. Music Conf.*, Copenhagen: 2007.

[12] A.R. Brown, "Exploring Rhythmic Automata," *Applications of Evolutionary Computing*, vol. Volume 3449, 2005, pp. 551-556.

[13] F. Pachet, "Rhythms as emerging structures," *Proceedings of 2000 International Computer Music Conference, Berlin, ICMA*, 2000.

[14] E.R. Miranda, "On the Music of Emergent Behavior: What Can Evolutionary Computation Bring to the Musician?," *Leonardo*, vol. 36, 2003, pp. 55-59.

[15] M. Dolson, "Machine Tongues XII: Neural Networks," *Music and Connectionism*, vol. 13, 1991.

[16] N. Tokui and H. Iba, "Music composition with interactive evolutionary computation," *GA2000. Proc. of the third International Conference on Generative Art*, 2000.