Laboratório VISGRAF
Instituto de Matemática Pura e Aplicada

Switching Interactive Modes

Sergio Krakowski
Luiz Velho (orientador)

Technical Report TR-2009-03 Relatório Técnico

January - 2009 - Janeiro

Sergio Krakowski Costa Rego
skrako@gmail.com

Rio de Janeiro
30.08.2008

Switching Interactive Modes

1 Introduction

This work is concerned with the problem of allowing the user of an interactive music system to switch between predefined modes of interaction without using external meta-commands. By external meta-commands we mean every information that cannot be coded strictly through the audio signal produced by the user during the interactive music experience (e.g. pedals, computer keyboard or mouse). Our approach to solve this problem is to use the rhythmic content of this audio signal in the form of *non-predefined rhythmic phrase commands.*
In [?] the author classifies interactive music systems using three dimensions:

1. Score-driven vs. Performance-driven systems.
2. Instrument vs. Player paradigms.
3. Transformative, Generative and Sequenced response methods.

Many composers applied score-driven methods to create human-computer interactive pieces (e.g. [?]).
Our general interest is to build perfomance-driven systems designed to enable musicians to build entire pieces of music (of about two and a half minutes) as the result of improvised interactive experience. In contrast to the score-driven paradigm, we are interested on solutions that allow the musician to make choices during the performance respecting this intrinsic characteristics of musical improvisation.
We can find many examples of performance-driven systems in computer music literature such as: [?, ?, ?, ?, ?, ?, ?, ?, ?, ?]. All of them present one or more modes of interaction that allow the musician to play with the computer to create an "improvisation" situation. These modes of interaction are predefined in the sense that the rules that govern the computer-musician interaction do not change during interaction. Even though some of them are designed to refine their interactive rules through machine learning procedures, their "objective" does not change during this learning process. Some of the cited works present more than one mode of interaction. In [?] a composition using a predefined sequence of these modes is presented, but the user don't have the option to change the order these modes appear during the interaction experience.

A limitation to all of these examples is the fact that the musician don't have the choice to change the mode of interaction he is using during the interaction experience without interrupting the music flow.
Our hipoteses is that a multi-modal system with a "natural" switching-mode strategy can offer the musician enough freedom to improvise during the computer experience but also a way to build non-monotonous pieces of music as is sometimes the case when using a one-mode interactive music system.
Thinking towards this direction we define mathematically the problem of switching modes of interaction and give our solution to it: the use of rhythmic content extracted from the audio generated by the musician as a command that don't interrupt the interaction experience. Because they are "inside" the musical information these commands can be considered more natural than the external meta-commands.
In the music field two examples can be cited as inspiration to this approach. The first one is the Bata trio. This drum ensemble is composed by a hierachy of three drums, the okonkolo (smaller drum), itotele (medium drum) and the yia (biggest leading drum). This percussion set is used in the Santerias ceremonies and different rhythmic patterns are played to different religious entities. The yia drum leads the other two drums. The yai player does that by play on of the possible "calls", rhythmic phrases that are interpreted by the other players as a sign to switch to another rhythmic pattern. The other example of "musical command" are the phrases used by the saxophone player Steve Coleman to give orders to his band. This incredible musician developed with his group a vocabulary of some phrase-orders that, when played, change the behaviour his band interacts with him. This can be heard in the piece "Colective Meditation (suite)" from the album "The Tao of Mad Phat" available on the web at **??**.
Taking into consideration the second dimension used by Rowe, the instrument versus player paragdigm, we locate our research on the instrument side. Because of that, we chose the automata theory as the mathematical formalism of our problem. This theoretic tool presented in section 2 is suited to model systems that react to command orders in oposition to the systems that try to make the computer emulate the human behavior in musical situations.
As an aplication to the modeling and resolution of our problem, we present in section 3, as a case study, a system designed for the interaction between a pandeiro (brazilian tambourine) player and a computer.
To prove the practical efficiency of this system, we present in section 4 two musical pieces built with it, a laboratory recorded piece and a live performance piece. Finally in section 5 we discuss the goals and drawbacks of our approach.

2 Mathematical Formalism

First we develop the formalism to deal with modes of interaction in general. Afterwards we focus on the formalization of rhythmic phrases that will be used to control these modes. Finally we give an overview of our problem and our solution to it using this formalism. We fix the notation of the set $\{1, 2, ..., n\}$ as $[n]$ throughout this work.

2.1 Modes of interaction

We chose the Automata Theory as the foundation of the concept of modes of interaction. As we are interested on studying interactive systems with many modes of interaction and as our main focus is on the interconnection between these modes, we need a model that deals with interconnected automata. We chose [**?**] as our main reference to this subject and we apply a simplification of his theory to our case.

Definition 1 *A deterministic finite automaton is a quadruple $A = (Q, \sigma, \delta, i_0)$, where*

> Q *is the finite set of states of* A
> $\sigma \in \Sigma$ *where* Σ *is the finite set of actions of* A *such that* $\Sigma \cap Q = \emptyset$
> $\delta \subset Q \times \Sigma \times Q$ *is such that if* (p, a, q) *and* (p, a, q') *are in* δ*, then* $q = q'$*; and*
> $i_0 \in Q$ *is the initial states of* A.

We can represent an automaton using an oriented graph such as in Figure 1.
We define a *mode of interaction* (also called *mode*) as a deterministic finite automaton. Each state Q represents a *musical context* (*context*) in which the mode can be. The actions $a \in \Sigma$ define the possible transitions from the current state to other states. During the interactive experience, the musician can choose (among some possible states) what state to go to and use this action to apply this state transition. Every time a change like that occurs, the mode produces a *musical answer* (*answer*) which depend on the current context and on the action chosen by the musician (this means we use a *Mealy machine model*, see [**?**]). The possible transition of states and the actions that cause these transitions is defined by the set δ. The initial state i_0 is the starting point of the musical interaction.
We are interested on studying interactive music systems with many modes of interaction so we need an object called *Synchronized Automaton.* Consider S the finite collection $\{A_i | i \in I\}$ of automata, where $A_i = (Q_i, \Sigma_i, \delta_i, i_{i0})$, and $I = [n]$.

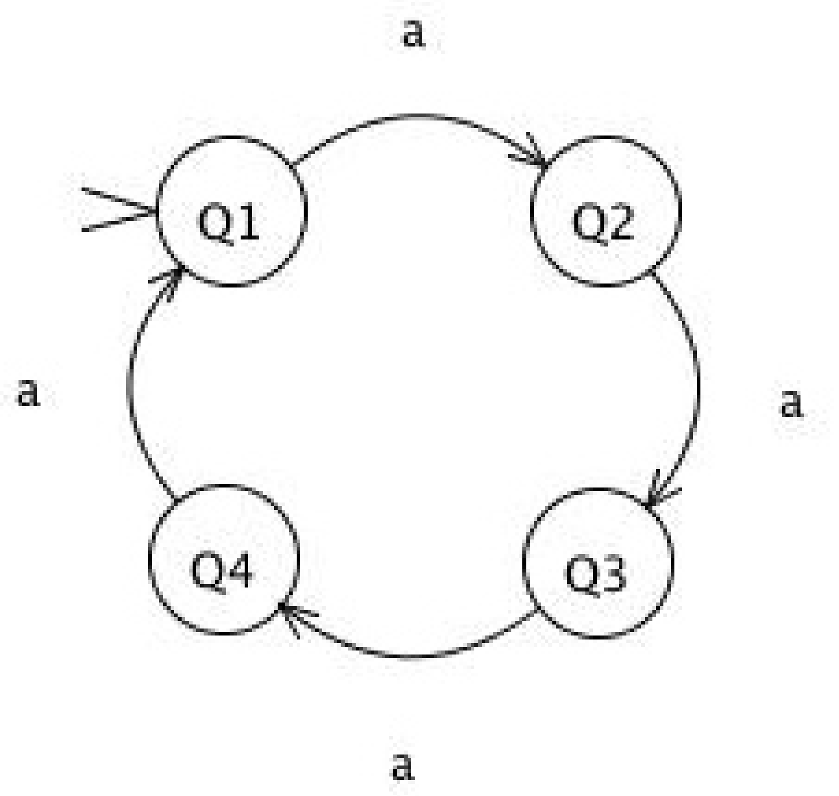


Figure 1: The vertices can be seen as the states of the automaton, and the letters in Q define the change of states. Here i_0 is signaled as a sign on the left side of the initial state.

Definition 2 *A synchronized automaton over S is a quadruple $\Im = (Q, \Sigma, \delta, i_0)$ where*

$$\begin{array}{l} Q = \prod_{i\in I} Q_i, \\ \Sigma = \cup_{i\ inI} \Sigma_i, \\ \delta \subset Q \times \Sigma \times Q \textit{ is such that } \forall a \in \Sigma, \\ \delta_a \subset \triangle_a(S), \textit{ and} \\ i_0 = (i_{01}, i_{02}, ..., i_{0n}). \end{array}$$

[**?**] gives the definition of the *complete transition space* $\triangle_a(S)$ where $a \in \bigcup \Sigma_i$, but we will omit it.
This construction of Q as the cartesian product of all the state spaces of each automata in the S collection gives freedom to create hierarchical structures of automata which is our interest here.
The general system we want to model has many predefined modes of interaction. We define them as the collection S above, each mode as an automaton A_i. Now we need a way to change from one mode to the other.
We define the *meta mode of interaction* (also referred to as *meta-mode*) as an automaton $M = (Q^M, \Sigma^M, \delta^M, i_0^M)$ in which each mode of interaction A_i is represented at least by one $q^M \in Q^M$. We call these meta-mode states as *meta-states*. The representation of each mode of interaction is defined by the relation r between the sets $\{A_i\}$ and Q^M.

Definition 3 *Given $r \subset \{A_i\} \times Q^M$, we say A_i is represented by q^M when $(A_i, q^M) \in r$.*

As said before, to each A_i, there exists at least one q^M such that $(A_i, q^M) \in r$.
Two other important concepts can now be defined.

Definition 4 *We say two modes of interaction A_i and A_j are excludent if $\forall q^M \in Q^M$, $(A_i, q^M) \in r \Rightarrow (A_j, q^M) \notin r$.*

Definition 5 *We say two modes A_i and A_j are* transparent *if we can listen to the audio output generated by both of them together. We say they are* opaque *if the audio generated by A_i is stopped when A_j outputs its audio, in which case we say A_j is* over *A_i.*

We can now define our system $\Im$ as the synchronized automaton over $S' = \{A_1, ..., A_n, M\}$. Furthermore, we need the $r \subset \{A_i\} \times Q^M$ as defined above.
We impose also the *representation constraint* over $\Im$. This constraint forces the mode of interaction A_i not to change its state if the meta-automaton M is in a state q^M such that $(A_i, q^M) \notin r$. Because of that, the relation r becomes crucial in the design of the meta-automaton and defines which automata can react in each meta-mode. This representation constraint can be defined rigorously in the following way:

$$\forall a \in \Sigma, ((s_1, ..., s_n, p), a, (s'_1, ..., s'_n, q)) \in \delta \Rightarrow \{s_i = s'_i \vee (A_i, p) \in r\} \forall i \in I.$$

(Examples!)

2.2 Rhythmic phrases

To control these modes of interaction, we propose the use of rhythmic information contained in the music signal generated by the user. Our assumption is that the rhythmic information is easily and quickly analyzed from a monophonic audio source with small amount of error and is the first step in a possible melodic/harmonic approach to be developed in future works.

[?] argues that the rhythmic content of an audio signal can be extracted only from the amplitude envelopes of each frequency band of this signal. Here we will deal only with the peaks of these amplitude envelopes which we will call *attacks*.
We formalize the idea of extracting information from the audio signal in the following definition:

Definition 6 *The sign r is called* automatically extracted from the audio signal *if there is an algorithm that calculates a feature value f_r (typically an average) over the digitized audio signal (typically divided in sample buffers) and outputs this sign r whenever this feature is above a certain threshold t_r. We call ϵ_r the probability of this algorithm to correctly extract the sign r from the audio signal.*

An attack is an example of a sign.
Also, we need mathematical tools to deal with the sequence of attacks detected during the interactive experience. As the length of this sequence cannot be defined in advance, we need a sequence of variable length.

Definition 7 *The* input sequence *s is defined as the function*

$$s : \mathbb{N} \times \mathbb{N} \to \mathbb{R}^+ \cup \{-1\}$$

such that given the current attack $c \in \mathbb{N}$,

1. $s(i,c) < s(j,c)\ ,\ \forall i < j \leq c \in \mathbb{N}$
2. *if* $i \leq c \implies s(i,c) \in \mathbb{R}^+$ *and if* $i > c \implies s(i,c) = -1$
3. $\forall k \leq c\ ,\ s(i,c) = s(i,k)\ ,\ \forall i \leq k$

We denote $s_c(i) = s(i,c)$.
We say that the sequence s_c is the input sequence defined until the current attack c and the positions $s_c(i) = -1, i > c$ are said to be undefined. Typically, $s_c(i), i \leq c$ is given in milliseconds.
Our interest is on the local information contained in part of the vector s_c.

Definition 8 *The* rhythmic phrase *(or just* phrase*) P_k of length k starting at position j of the input sequence s is defined as the vector*

$$P_k = (s_c(j), s_c(j+1), ..., s_c(j+k-1)).$$

For the sake of clarity we can make explicit the starting position j, the input sequence s and the vector index i in the notation $P_k(j,s)(i) = P_k(i)$ but we try to avoid that, not to overload notation. It's important to notice $P_k \in (\mathbb{R}^+)^k$. Sometimes we also omit the k and write just P.
Finally we define a distance between two phrases with the same number of attacks.

Definition 9 *The distance $d : \mathbb{R}^k \times \mathbb{R}^k \to \mathbb{R}^+$ is defined as*

$$\left|\frac{l(P_k^1)}{l(P_k^2)}\right| + \Sigma_{i\in[k]}\left|\frac{P_k^1(i)}{l(P_k^1)} - \frac{P_k^2(i)}{l(P_k^2)}\right|$$

$\forall P_k^1, P_k^2 \in \wp_k$ where $l : \mathbb{R}^k \to \mathbb{R}^+$ is the length function

$$l(P_k) = |P_k(k) - P_k(1)|$$

of phrase P_k.

2.3 The problem

Now our problem can be stated in the following way:

Problem 1 *Given $\Im$ synchronized automaton as defined above where:*

1. *equation* **??** *is satisfied,*
2. *the modes $\{A_i | i \in I\}$ are excludent, and*
3. *$\forall i \neq j \in [n], \exists a_j \in \Sigma^M$ such that*

$$(r(A_i), a_j, r(A_j)) \in \delta^M.$$

Our problem is to associate to each a_j a sign r_j automatically extracted from the audio signal. We abuse the notation and say this sign switches *from the current mode A_i to the mode A_j, or that this sign r_j* is interpreted as *action a_j.*

When this problem is solved, the musician can use the sign r_j to switch the current mode of interaction. As r_j is automatically extracted from the audio signal, the user doesn't need to use any external meta-command to control the meta-automaton.

Our solution to this problem is to choose n different phrases $\{P_k^j | j \in [n]\}$ and to define the sign r_j as the detection sign whenever this j-th phrase is played by the user.
The algorithm that detects if the phrase P_k^j has been played is very simple: it checks using the distance function d if the last k-length phrase $(s_c(c-k+1), s_c(c-k+2), ..., s_c(c-1), s_c(c))$ in the input signal is sufficiently close to P_k^j.
Two other important signs are defined here. They are used in the case study presented in this paper to solve many important interactive tasks.
The first one is the *detection of phrase repetition*. This means to detect if the user repeated a certain phrase P_k twice contiguously in time. The way to detect this is close to the way used to detect if a pre-defined phrase has been played. The algorithm checks if the phrase $P_k = (s_c(c-2k+1), s_c(c-2k+2), ..., s_c(c-k-1), s_c(c-k))$ is sufficiently close to $P'_k = (s_c(c-k+1), s_c(c-k+2), ..., s_c(c-1), s_c(c))$, but in this case the k must variate. Typically, the algorithm checks for the distance $d_k = d(P_k, P'_k)$ for $k = 2, ..., 20$ and if for one of these k's d_k is smaller than a certain threshold, it generates a sign that P_k has been repeated. If both d_k and $d_{k'}$ are bellow this threshold, the algorithm gets the $argmin(d_k, d_{k'})$. This sign is different from the other signs defined until now. The algorithm that outputs it can also output the phrase P_k that has been repeated. Until now the signs could represent only a binary information which would be if the detection of a certain event happened or not, the repetition sign also *carry* information about what phrase has been repeated. That is the reason this sign is used for many purposes in the case study presented here.
The other important sign is the *silence sign*. It is generated if the user stops playing for a certain amount of time. Although it is a simple sign, it can be used for many purposes e.g. to segment long phrases or to change the mode of interaction in a sequential meta-mode framework.

3 Pandeiro Case Study

We discuss here a case study of a system designed to interact with the Pandeiro (brazilian tambourine) that has three modes of interaction and uses the solution defined above. Two of these modes uses the Continuator system (**??**) to produce harmonically interesting piano chords. All three modes and the meta-mode that controls them is now described in details.
A presentation of the Pandeiro and the low level analysis used to generate the attack signs can be found in **??**. First we describe the architecture of the system using the tools already defined. Then we show a graphical representation of two musical pieces developed using these systems and comment on the resemblances and differences between them.

3.1 System Architecture

The system has been built using the Pure Data framework that fits well to real-time interactive applications (**??**). We use just the basic functionalities of this software such as audio acquisition and buffering, time management, data recording and output of midi information. All the core elements responsible for the interactive tasks of this system have been implemented in C as pure data *externals* (term given to the objects that are not part of the set of basic functions).
The three modes of interaction are described here as well as the meta-mode that controls them.

3.1.1 Mode 1

The first mode of interaction is a user-programmable drum machine. The concept of drum-machine dates back to the 30's and became popular in the 80's with the Roland TR-808 machine. The purpose of these machines was to create drum kit grooves using synthesized or sampled sounds. The basic idea is to divide the drum loop in voices. Each voice can be seen as a separate one-instrument loop that is summed up to form the whole groove.
Mode 1 allows the player to build a whole drum-machine loop by programming each voice without external meta-commands. The musician uses the phrase repetition sign to do that. In this case study, the instruments chosen to be the loop's voices were: the bass drum, the snare drum, the hi-hat and the iron triangle.
The automaton that represents this mode is depicted in figure 2. We call this the automaton $A_1 = (Q^1, \Sigma^1, \delta^1, i_0^1)$. As we see in that figure, this automaton has four states $\{Q1, Q2, Q3, Q4\} = Q^1$ and just one action $\{a\} = \Sigma^1$. This action is associated to the repetition signal previously defined. Each time this sign is received, the automaton leaves a certain state and builds one of the voices of the drum machine loop.
We now explain how it works. In the initial state $Q1$, the automaton doesn't produce any audio output. If the user repeats whatever phrase P^1 twice, the algorithm that detects repeated phrases in the input sequence will generate a repetition sign which makes the automaton switch to state $Q2$. When it leaves state $Q1$ it produces the bass drum voice. This voice is an audio signal that is looped many times (further on we explain when this loop stops). Figure **??** explains how does this audio is constructed. As we see in that figure, the audio signal is just a copy of phrase P^1 but instead of the original pandeiro sounds there is a bass drum sampled sound placed at the attack instants of this phrase.
The automaton is then in state $Q2$ and is prepared to record the snare drum voice. When the user repeats another phrase P^2 the automaton changes to state $Q3$ and generates the snare drum voice exactly as happened in the bass drum case. The bass drum and the snare drum voices are summed up and the user can listen to both of them. It's important

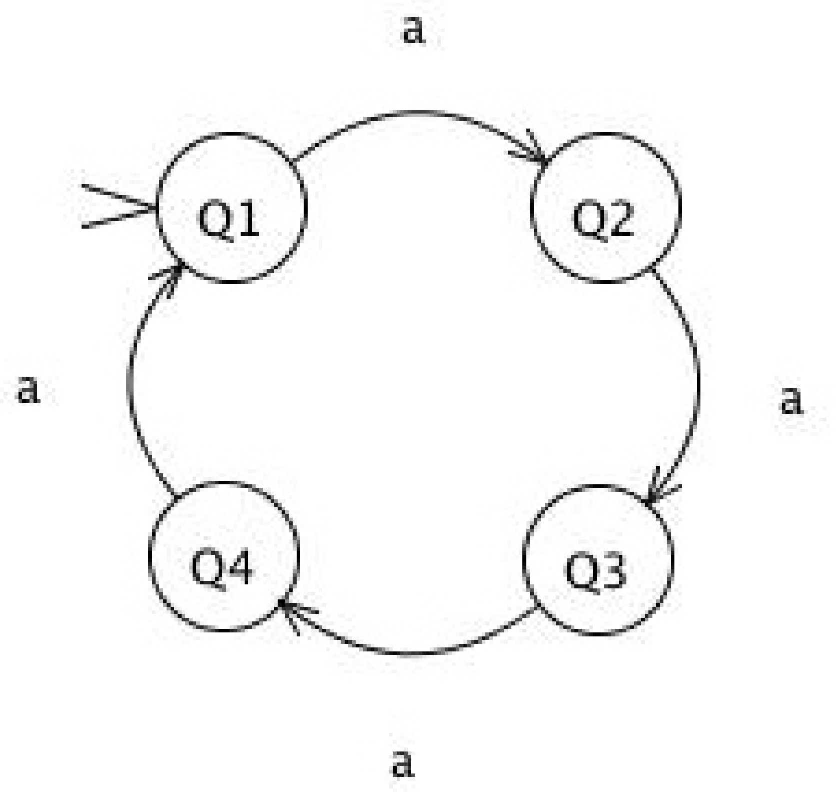


Figure 2: Each state represents a voice to be recorded. Every time a phrase is repeated, the automaton starts to loop this phrase and changes its state to be prepared to record the next voice.

to notice that the voices need not to be of the same length. This allows the user to create polyphonic effects as depicted in **??**. The algorithm that manages the building of the voices adjusts their length not to occur a desynchronization phenomenon.
Again, when the user repeats another phrase P^3, the automaton leaves state $Q3$ to state $Q4$ and produces the hi-hat voice. If the user repeats the fourth phrase P^4, the iron triangle loop will be recorded and the automaton will go back to state $Q1$. At this moment, the bass drum previously recorded is deleted and the user can record it again. When the user passes to state $Q2$ the new bass drum is recorded, the snare drum voice is erased and so on. This way the user can progressively change the content of each voice which creates a dynamism of the drum machine groove.

3.1.2 Mode 2

This mode of interaction permits the user to release short duration (0.2s long) piano chords simultaneously to each attack produced by the pandeiro.
The chords are not produced by our system directly, instead, they are produced using the*Continuator system* in the harmonizer mode. In this mode, the Continuator receives a note (using the MIDI protocol) and chooses the best chord (also represented using MIDI) that harmonizes it. This judgement is done with respect to the musical style the system learnt from the player in a previous stage. As the Continuator uses this style dependent criteria to harmonize each note, it's possible to listen to a harmonic coherence in the sequence of chords released by the pandeiro attacks in the musical examples of this case study. That was the motivation to use the Continuator instead of, e.g., a random choice of chords.
Our system have to decide what note to send to the Continuator at each pandeiro attack. In this case study we used two strategies of choice (one to each piece here presented): a pre-defined sequence of twelve notes and a pseudo-random choice in a chromatic scale.
First we present the strategy of the pre-defined sequence. The automaton A_2 used to model this choice is depicted in figure 3. Each state $R1$, $R2$, ..., $R12$ represents a note of this pre-defined melody. The action b is associated to the attack sign. This means if the automaton is in state $R1$ and the player produces an attack sign, the automaton sends note 1 to the Continuator to be harmonized and goes to state $R2$. The chord generated by this system is sent back using MIDI information. This information becomes audio output using a Hammond piano synthesizer. Ideally, this synthesized chord should be heard at the same instant we can hear the pandeiro attack, as if they were "attached" to each other. In our case, there is a small delay (around 30 ms) between the attack and the chord release that is barely perceptible and, thus, can be ignored.
Then, when the system is in state $R2$ and the user produces another attack, the second note is sent to the Continuator, harmonized and a new chord is synthesized again. The automaton goes to state $R3$ and so on. When it arrives at state $R12$, it goes back to state

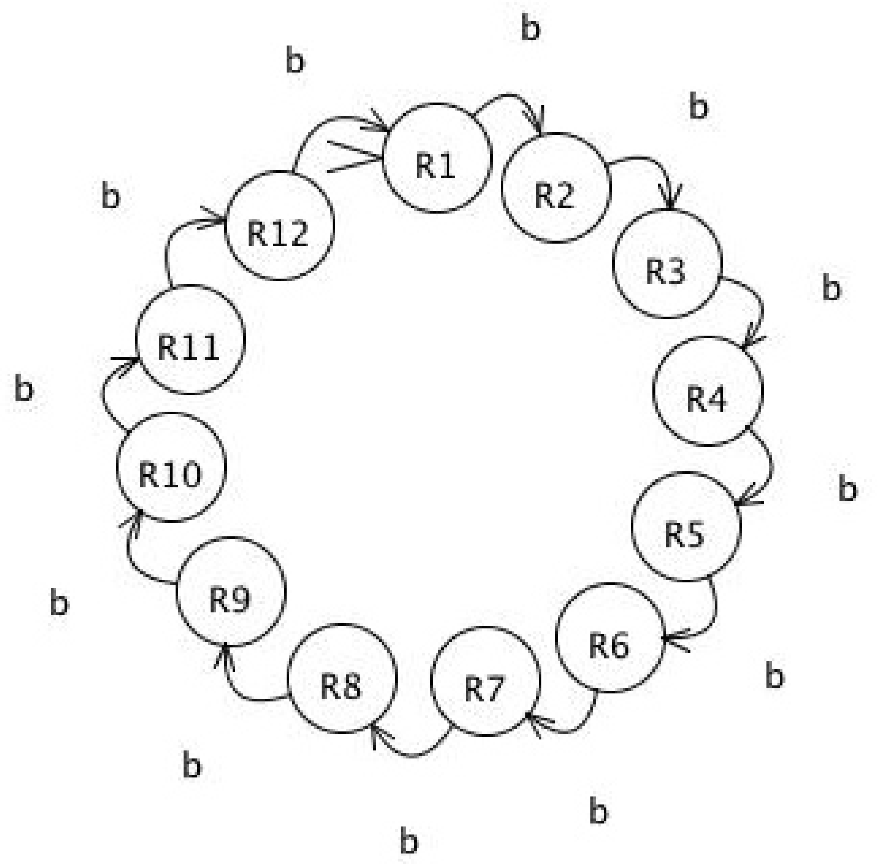


Figure 3: Each state represents a note to be played. At each attack sign, the automaton plays this note and goes to the next state.

one and restarts this twelve notes cycle. Although the melody sent to the Continuator is cyclic, the harmonization changes each time this cycle is repeated, which gives a richness in terms of harmonic paths, even though the sense of cyclic melody is still preserved. This is another motivation to use the Continuator as harmonizer. All that will be observed in the examples presented further on.
The second strategy is to choose randomly whether to go up or down in a chromatic scale. We omit the figure that represents this automaton, but it suffices to say it has only one state and one action b that links this state with itself. Every time the user produces an attack sign, the automaton sends a note to the Continuator and comes back to this state. This note can be either one semitone higher (.5 of probability) or lower (.5 of probability) than the note sent in the last attack sign. As will be observed, in this case, the melodic sense is absent, but the harmonic coherence is still preserved.

3.1.3 Mode 3

This mode is close to Mode 2. It allows the user to release a long duration (2s) piano chord.

Figure 4 shows the automaton A_3 that models this mode. As can be seen, there is only one state $S1$ and the action c. This action is associated to the detection of a certain phrase P^2. Further on we will explain how does this phrase is defined. Every time this phrase is detected in the input sequence, our system sends a C note (MIDI note number 72) to the Continuator which harmonizes it. This chord is synthesized as before, the only difference is that the duration of its sound is two seconds long instead of the two hundred milliseconds of Mode 2. The other difference from this Mode to Mode 2 is that the chord is played “immediately” (the latency can again be ignored) after the last note of phrase P^2 is detected, in the previous case, the chord was released after each attack was detected.

3.1.4 Meta-mode

The role of the Meta-mode is to model the meta structure that allows us to work with all these modes of interaction in the same musical system. In figure 5 we depict the automaton that represents this meta-mode. As we exposed before, our problem was to be able to switch from one mode of interaction to the other and the solution we proposed was to use rhythmic phrases for this purpose.
Our system will be a practical example of this solution. An important characteristics of it is that these phrases are not predefined, this means the user is able to choose them in a setup phase previous to the musical interaction itself.
This phase is represented here by the states $M1$, $M2$ and $M3$, and they will be responsible for the recording of the phrases P^1, P^2 and P^3 respectively.

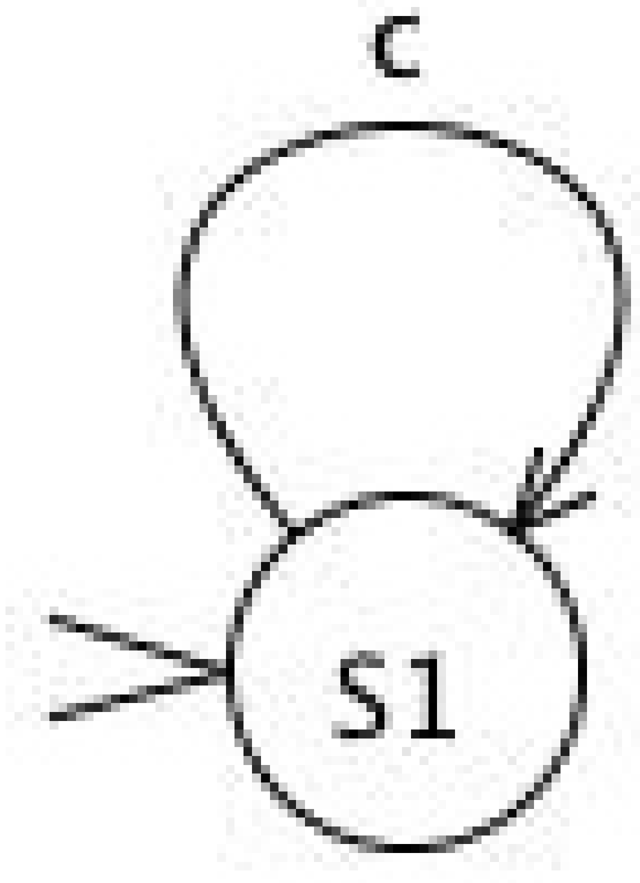
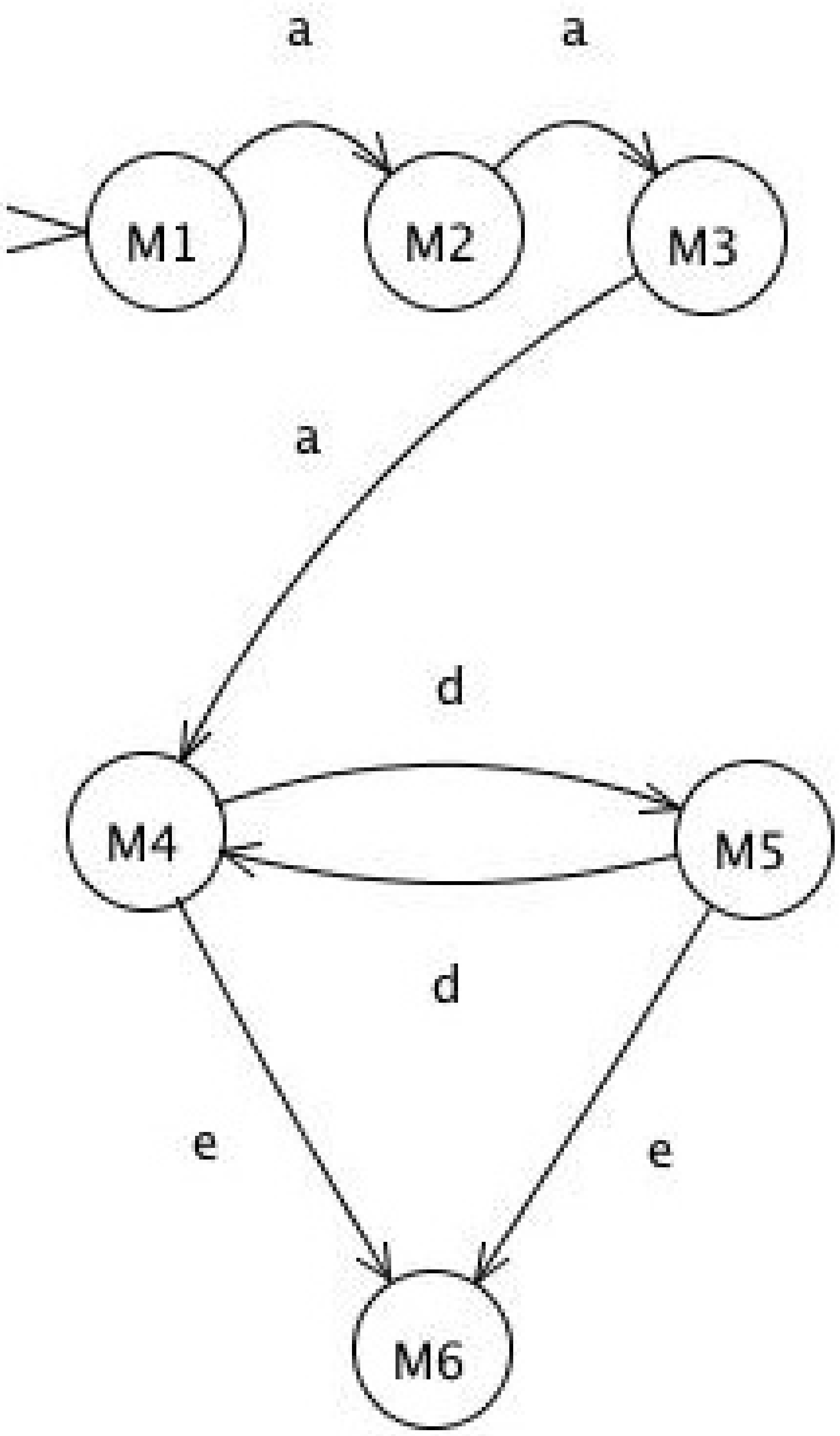


Figure 4: Each time a c action is applied, the automaton releases a long chord and comes back to its unique state.



$$\begin{tikzcd}
\arrow[r] & M1 \arrow[r, bend left, "a"] & M2 \arrow[r, bend left, "a"] & M3 \arrow[dll, "a"'] \\
& M4 \arrow[rr, bend left, "d"] \arrow[dr, "e"'] & & M5 \arrow[ll, bend left, "d"] \arrow[dl, "e"] \\
& & M6 &
\end{tikzcd}$$

Figure 5: Meta-automaton that controls the behaviour of the other automata.

We notice the action a is found again in this automaton. This action a is also associated to the repetition sign as before, but here its utility will be different from the drum machine case. When the meta-automaton is in the initial state $M1$, if the user repeats a certain phrase, this phrase is recorded as P^1 and the meta-automaton changes to state $M2$. No musical answer is generated at this transition, that's why we call it setup phase. Again the user repeats another phrase which is recorded as P^2 and the meta-automaton passes to state $M3$. Finally the last phrase P^3 is recorded in the same way, the meta-automaton leaves the setup phase and passes to the meta-mode $M4$.
The meta-automaton enters the interactive phase (states $M4, M5$ and $M6$) that will be described in details later on. For now, it suffices to say that Mode 1, the drum machine mode is represented only by the meta-mode $M4$; Mode 2, the short duration chord release mode, is represented only by $M5$; and Mode 3 is represented by both meta-modes $M4$ and $M5$. This means Mode 1 and Mode 2 are excludent, the user can play with one or the other, but Mode 3 is active when any of them is active, the user can access it while playing with either one of the other two modes.
The action d is associated to the detection of phrase P^1. This means the user can go from meta-mode $M4$ to meta-mode $M5$ and backwards just by playing this phrase. Action e is associated to the detection of phrase P^3, whenever the user plays it the meta-automaton will go to the end state, the musical piece will finish. As seen before, the detection of phrase P^2 is associated to action c, which causes the automaton A_3 to release a long duration chord if the meta-automaton is in the state $M4$ or $M5$.

Actions of automaton $\Im$	
action	sign
a	repetition sign
b	attack
c	detection of phrase P^2
d	detection of phrase P^1
e	detection of phrase P^3

As we can see, the three phrases above mentioned are recorded in the setup phase and the detection of each one of them is associated to the actions d, c and e respectively. This is summarized in table **??**.
Now we can describe the synchronized automaton $\Im$ over $\{A_1, A_2, A_3, M\}$ which will model the whole system. We chose to represent this automaton graphically in figure **??**. As the state set $Q^{\Im}$ is in the cartesian product $A_1 \times A_2 \times A_3 \times M$, we will represent a state of this automaton as a red dot inside one state of each of these automata depicted in this figure.
In table **??** we present all the possible transitions of this automaton.

Transitions of a

transition n^o	initial state	action	end state	comments
1	$(Q1, R1, S1, M1)$	a	$(Q1, R1, S1, M2)$	
2	$(Q1, R1, S1, M2)$	a	$(Q1, R1, S1, M3)$	
3	$(Q1, R1, S1, M3)$	a	$(Q1, R1, S1, M4)$	
4	$(Q(n), R1, S1, M4)$	a	$(Q(n+1mod4), R1, S1, M4)$	$\forall n \in [4]$
5	$(q, R1, S1, M4)$	d	$(q, R1, S1, M5)$	$\forall q \in Q^1$
6	$(q, R(n), S1, M5)$	b	$(q, R(n+1mod12), S1, M5)$	$\forall q \in Q^1, \forall n \in [12]$
7	$(q, r, S1, M5)$	d	$(q, R1, S1, M4)$	$\forall (q, r) \in Q^1 \times Q^2$
8	$(q, r, S1, m)$	c	$(q, r, S1, m)$	$\forall (q, r) \in Q^1 \times Q^2, m \in \{M4,$
9	$(q, r, S1, m)$	e	$(q, r, S1, M6)$	$\forall (q, r) \in Q^1 \times Q^2, m \in \{M4,$

When the context allows, we omit the state of $\Im$, which is a cartesian product, and talk about the state of the other automata A_1, A_2, A_3 and M. (E.g. the transition $(Q1, R1, S1, M1)$ to $(Q1, R1, S1, M2)$ is simplified as $M1$ to $M2$).
The transitions $1, 2$ and 3 are part of the setup phase. The user can only skip from $M1$ to $M2$, then to $M3$ and finally to $M4$ using the repetition sign as described before. The other automata remain idle, all of them in their initial state.
In the meta-mode $M4$ the user have four options:

1. to use action a and program one voice of the drum machine (transition 4 of table **??**),
2. to change to meta-mode $M5$ using action d (transition 5),
3. to release a long duration chord using action c (transition 8), or
4. to finish the piece using action e (transition 9).

The user can choose a repeatedly and program as many voices of the drum machine as he or she wants.
If the user chooses to go to the meta-mode $M5$, then the drum machine stops being programmed. This doesn't mean the drum loop already programmed stops being played. In fact the Modes 1 and 2 are transparent, so the audio result of both can be heard at the same time. In meta-mode $M5$, each attack is read as action b (transition 6) which will release a short duration chord. The user can go through the twelve-note melodic cycle as much as he or she wants. In this situation the user can also:

1. come back to the meta-mode $M4$ using d (transition 7),
2. release a long chord using c (transition 8), or

3. finish the piece using e (transition 9).

The action c generates the long chord release in both meta-states $M4$ and $M5$. Apparently the automaton $\Im$ stays in the same state when this action is applied. For example: consider $\Im$ is in state $(Q1, R1, S1, M4)$ and the user plays phrase P^2. The final state of this transition (n^o8) will be again $(Q1, R1, S1, M4)$, but $\Im$ has described the cyclic arch of automaton A_3 which means the musical answer to c will be heard, in this case, a long duration chord release. It's important to notice that during the two seconds this chord sounds, the audio from any other Mode is muted. This means Mode 3 isn't transparent with respect to Modes 1 and 2, but is over them.
We can also observe in table **??** all the answers and computational consequences that happen when the user chooses a certain transition. Some of the answers are musical and others are just functional.
We give now an example on how a user can play with this system. The initial state will be $(Q1, R1, S1, M1)$. The only option is to repeat a certain phrase (action a) and go to $(Q1, R1, S1, M2)$. This has to be done twice and the user goes to the state $(Q1, R1, S1, M4)$ having recorded all the three phrases used as controls. The user programs the bass drum voice by repeating another phrase (action a). This voice starts to be looped and the user can create the snare drum voice by playing over the bass drum voice (action a). Then the user decides to go to the short chord release mode by playing phrase P^1. At each attack, one of the twelve chords is released. b is repeated, let's say, six times. Then the user plays again phrase P^1 (which means the other k_1 attacks are played and generate more k_1 short chords) which takes him or her back to the drum machine mode. Now the user can program the hi-hat (note the automaton A_1 was idle and stayed in state $Q3$). The musician applies a twice again programming the hi-hat and the iron triangle voices. The bass drum voice is erased. The user applies c and after the drum machine loop restarts to sound. Finally he or she decides to finish the piece and apply e. All this example could be summarized in the sequence of actions $(a^5, d, b^6, b^{k_1}, d, a^2, c, e)$. In fact the representation used to describe the two following musical examples will be given by a timeline tagged with each action performed by the user at the instant they took place.

(Do a real example!!!) (The meta-mode do not generate audio! The modes do.)

4 Musical Examples

As we let clear in the introduction section, our final interest is to be able to create entire pieces of music using a performance-driven system. Here we present two pieces to pandeiro and computer, performed using the system described in our case study. This proves the problem we addressed and the solution we gave to it in this report point towards our final interest.

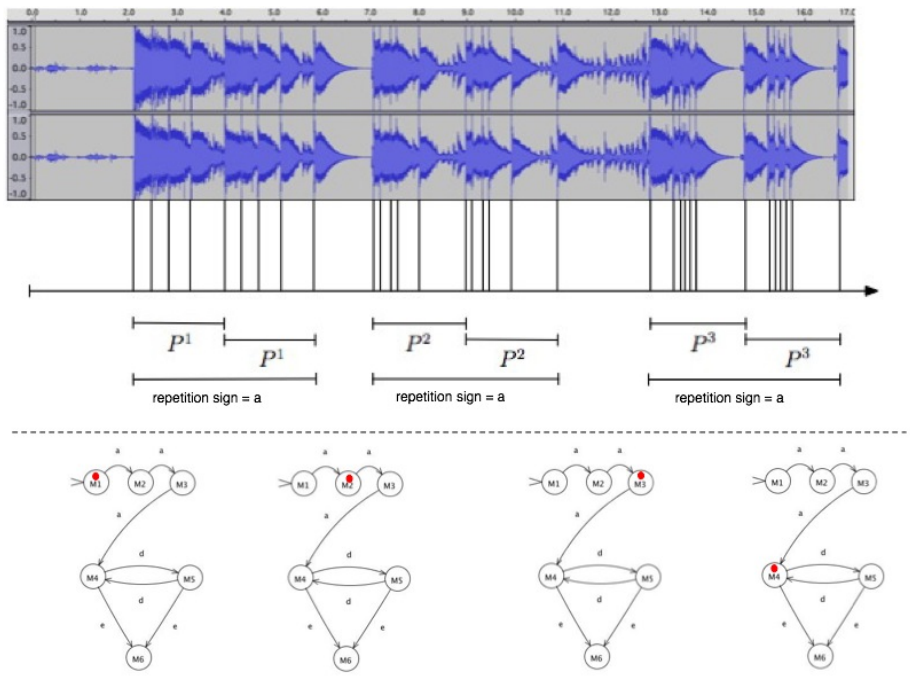


Figure 6: Above the dashed lines we see the audio wave and the attack signs grouped in phrase P^1, P^2 and P^3 respectively. The repetition of each phrase is the sign associated to action a. Below the dashed line we see the state transition of the meta-automaton caused by this action each time the repetition sign is detected.

In this section we will present both pieces and discuss the main goals and drawbacks of our approach.

4.1 Laboratory Piece

This piece has been developed at the Sony Computer Science Laboratory in Paris during a long section (about an afternoon) in which the pandeiro player, first author of this report, used the system described above. All the section is recorded in video and is available in **??**. The recording is divided in 24 clips of about three minutes each. The piece had not been composed before this section, instead, it has been built up during the many trials the musician did while playing with that system. We won't comment all these 24 clips here but it is possible to notice, in the first clips, the player gradually defines the piece (until clip 10) and then makes several adjusts (clip 11 to 24) to perfect it. Here we will describe only one clip (**??**) where the piece is already settled. Our aim is to make a detailed description of all the clips will be done in the future.
Before describing the piece it's important to present the setup phase the user had to go through as explained before. The video **??** shows this phase. We extracted its audio and

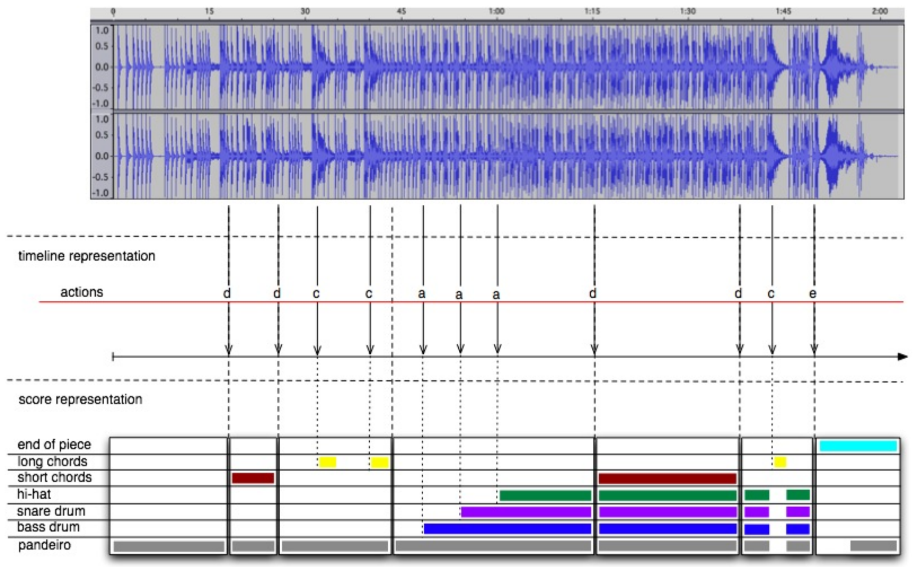


Figure 7: We see three representations in this picture. The audio wave, the timeline representation showing the instant each action is applied and the score representation showing a scheme of each of the seven parts of the piece.

depited in figure 7. We can see the three phrases being recorded and the state transitions of the meta-automaton that control this recording.
From this point, the user could start to use the system as mentioned above.
We depict a graphical representation of the piece in figure **??**. In the top of this figure, we see the audio wave extracted from the video clip **??**. Below that we see a timeline representation of the piece that shows when each sign has been detected and caused the automaton $\Im$ to change its state. Finally we see a schematic score representation where the piece is divided in seven parts. In this last representation, we see when each “voice” can be heard during the piece. By “voice” we mean all kinds of sounds heard in the audio wave, this means the pandeiro played by the musician during the piece, the drum sounds generated by the computer (bass drum, snare drum and hi-hat), the harmonic sounds also generated by the computer using the continuator (short chords and long chords) and finally the group of sounds that signal the end of the piece (some chords followed by a cymball sound).
Now we can describe in details what happens during the piece. In the beginning, the automaton $\Im$ is in the state $(Q1, R1, S1, M4)$ (notice the setup phase is finished and the automaton is already in the interactive phase). The first part of the piece is a pandeiro solo. In the end of this part, the musician plays phrase P^1 which is detected and interpreted as action d. This causes $\Im$ to change to state $(Q1, R1, S1, M5)$.
Action d marks the beginning of part two. As we know, in this state $(Q1, R1, S1, M5)$ each attack is interpreted as action b (which releases a short chord) but it’s important to notice

these actions were omited in figure **??** for the sake of visual clarity. At this point, the musician plays twelve attacks completing the whole melodic cycle mentioned before. At each attack, $\Im$ goes from $(Q1, R1, S1, M5)$ to $(Q1, R2, S1, M5)$, then to $(Q1, R3, S1, M5)$ and so on until reaches the state $(Q1, R12, S1, M5)$. The last four attacks form phrase P^1 which is interpreted as action d again. This action takes $\Im$ back to state $(Q1, R1, S1, M4)$ and marks the beginning of part three (notice action d takes wathever $R(n)$ to $R1$ as defined in transition n^o7).
In this part, the musician improvises a bit more of pandeiro solo and plays phrase P^2 which is interpreted as action c. This causes $\Im$ to describe an arc transition from $(Q1, R1, S1, M4)$ to itself which releases a long chord. Again the player improvises a bit more and plays P^2 releasing another long chord. This marks the end of part three.
In part four the player repeats twice a certain phrase and this is interpreted as action a. The automaton $\Im$ builds the bass drum voice as a looped copy of this phrase and goes to state $(Q2, R1, S1, M4)$. The bass drum voice can now be heard and over that, the player repeats another phrase which becomes the snare drum voice. Both voices are being played by $\Im$ which is now in state $(Q3, R1, S1, M4)$. Then the player repeats the last phrase that becomes the hi-hat voice. At this moment, $\Im$ is in state $(Q3, R1, S1, M4)$ and these three voices form the drum machine groove. The player improvises over this groove and finishes this part by playing phrase P^1.
We notice until this moment, the piece increased in terms of musical complexity: the beginning is a pandeiro solo followed by some chords and the progressive construction of a drum machine loop. In part five ($\Im$ is at state $(Q4, R1, S1, M5)$) the piece reaches the peak of this complexity because it's the moment all the drum layers are on and the player exposes the twelve-note melody three times. At each exposition the player uses a different rhythmic placement of each note creating variations of the melody. The last four attacks of this part form phrase P^1 which turns off the chord release mode ($\Im$ goes to state $(Q4, R1, S1, M4)$) and marks the beginning of part six.
In this part the drum machine groove is still on and the player improvises again over this base. It's important to notice the player cannot repeat another phrase unless he wants the computer to record another drum machine voice (the iron triangle voice). This voice has not been recorded in this piece. In fact, at that moment, the player had to pay attention not to repeat a phrase, which constrained his musical freedom (we will discuss more about that later). During this improvisation, the musician played phrase P^2 interpreted as action c causing the release of a long chord. This is an important instant of the piece because this chord release muted the whole drum machine loop (this can be seen in the score representation as a gap in the three drum voices). As the player also stayed in silence during the two seconds this chord sounded, a musical tension has been created. This tension is solved with the return of the drum loop and the pandeiro sounds. Finally, to end the piece, the musician plays phrase P^3 interpreted as action e. This makes

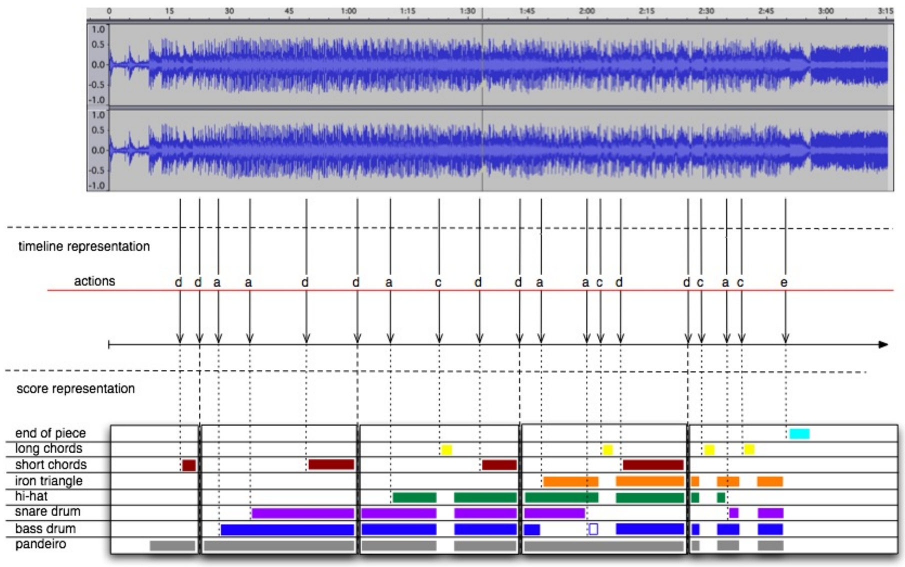


Figure 8: The same graphical representation is applied to this piece.

the system mute all the audio output it was generating (in this case the three drum voices being looped) and produce a predefined sequence of sounds that end the piece. This sequence consists of three chords followed by a cymball sound. For the sake of clarity we defined part seven as the release of this sequence together with the musical answer given by the pandeiro player which is a typical "fade out" end. In this part, the automaton $\Im$ is at the final state $(Q4, R1, S1, M6)$ and the interaction is already finished because there is no possible action defined in this state.

4.2 Live Piece

The other piece presented here has been performed live at the Oldham Percussion Festival by the first author of this report and recorded on video available at 8. The system used to this performance is slightly different from the one presented before. In this case the setup phase has not been recorded on video and the phrases associated to the automata actions are not the same. The other difference is the short chord mode which uses the chromatic strategy to send the notes to the *Continuator* as we described before.
The piece is represented in figure 8 as before. We omit its whole description but we point out some important details that happened here.
In this case, the piece is divided in five parts. It is clear from the recording that, differently from the first one, this piece has been built in an improvised way (graphically we can see this piece is less "organized" than the first one). The player had in mind a background structure: the alternation between the drum machine mode and the short chord mode but some choices were made on stage during the performance, such as when to release

long chords, the content of each drum machine voice and when to finish the piece.
A characteristics of improvisation is to deal with the unexpected. In this piece, uncorrect detection of certain signs caused unpredicted behaviour of the machine that will be commented further on.
As we can see in figure 8 each part (except the last one) finishes when the player leaves the short chord mode represented as red boxes in the score representation. This was the criteria used to partition the piece.

In general terms, this piece bear some resemblance to the first one. As we see graphically, the musical complexity also increase from the beginning to reach at part four its peak.
In this case, more drum machine voices were built. In the score representation we see the first two drum voices were created in the second part (bass drum represented as a blue box and snare drum represented as a magenta box), then one more (hi-hat represented as a green box) is created in the third part. The iron triangle voice, represented as an orange box, is created in the fourth part. Notice when this voice is created, the bass drum voice is stopped. As we wrote before, when the drum machine goes from state $Q4$ to $Q1$ the bass drum voice is erased and this automaton is prepared to record it once again.
In part four the bass drum voice is recorded again and the snare drum stops, but something unpredicted happened at that moment. If we look at the score representation we see two actions a were taken in this part. The bass drum is recorded in the second a action but right after this recording the user accidentally plays phrase P^2 interpreted as a c sign which causes the release of a long chord (yellow box). As we saw before, the long chord mode isn't transparent and is over the other modes, so the audio from the drum machine is muted and we can realize the bass drum has been recorded only after the long chord stopped sounding. That's why we represented the beginng of this bass drum voice as an unfilled blue box. Also, the snare drum voice has been erased for the same reason explained above. Finally, another snare drum voice is recorded in part five.
The short chord mode also appeared more often in this piece. As we saw before, here we don't have a melodic sense in the notes sent to the *Continuator*, but we can listen to the harmonic coherence this system gives to the sequence of chords. The first action d leading to mode 2 was not intentional. The first part of the piece was supposed to be a pandeiro solo where the player was supposed to do an *accelerando* to reach the tempo of the piece, but, during that, he played phrase P^1 which led him to the short chord mode. As that was not intentional, the player soon left this mode to start recording the drum voices (part two). It's possible to notice the player smiled when that happened, because it was unexpected.
The c actions were concentrated in the end of the piece. As we saw before, the second c action was unexpected, and the user didn't stop playing during the long chord sound, but the other three c actions were followed by a pandeiro silence which created tension as

in the first piece. The end action e has been correctly interpreted.
The unpredicted detection of those two actions (the first d of the piece and the second a of part four) didn't compromise the performance and we can tell by the intensity of applause the audience seemed to like it.

5 Conclusions