Interactive scores : A model for specifying temporal relations between interactive and static events

M. Desainte-Catherine, Antoine Allombert

Interactive Scores: a Model for Specifying Temporal Relations Between Interactive and Static Events

Myriam Desainte-Catherine Antoine Allombert
SCRIME, ENSEIRB, LaBRI, université Bordeaux 1
351, cours de la Libération 33405, Talence Cedex, France
myriam@labri.u-bordeaux.fr allomber@labri.u-bordeaux.fr

Abstract

We propose a formalism for specifying temporal relations between interactive triggerings and releasings occurring during performance of written musical pieces. Temporal durations are specified between parts or notes of a written piece. Then, we proceed to a static analysis of the piece in order to produce a program providing safe execution of the piece according to the temporal relations.

1 Introduction

Actual computers allow various kinds of real time interactions with sound synthesis. The huge quantity of possible ways of interaction in contemporary music shows the great diversity of new possibilities provided by computers, but also the difficulty of aiming to formalize and generalize these practices. The recent research on sound interaction focus on the analysis of the gestures of the musician and the mapping of the parameters of these gestures with the parameters of the sound synthesis. Some research deal with musical associations (harmonic and melodic organization), but none has been carried out in order to propose a general model. We think that such a model would help to unify the different ways of interaction of musical and sound levels and bring a better understanding of musical interaction.

Actual tools for writing musical scores take into account interactive events in a very limited way. The only way for the composer to control precisely the musical piece resulting from real-time interactions with a computer, is to describe the calculus binding his musical objects by the means of a programming language. Our aim is to provide the composer a way to specify properties, rather than calculus, and to compute automatically the resulting musical piece from its specification. In this paper, we consider that the composer action is limited to organize some notes in time. We prefer this quite primitive concept to provide an intuitive tool to composers and musicians who have not necessarily computing skills. We can note that actual systems and models which take into account interactivity such as the HARP system [Camurri et al.1995] need to be use acurate understanding of complex computing concepts. Even if a lot of actors of contemporary creation deal with shuch concepts, we want to center our model around the musical language to allow composers to start their creations from musical aspects rather than computing aspects. We also want the model to be used for pedagogical applications by people who absolutly not deal with computing concepts. Another fact is that we want a generic tool. We met systems designed for one creation (musical piece or multimedia happening) but we want to separate musical pieces from computing aspects.

We start this study by focusing on temporal aspects of interaction, putting aside, for now, sound synthesis. So as a begining we don't take care of the musical parameters but only of the dates of begining and end of sounds. But we are aware of the compatibility of our model with actual systems using continuous controls. We start with a piece which is completely composed and we want to provide the composer a way to define a set of possible interpretations of his piece by specifying simple temporal constraints between musical objects. Allen's temporal relations [Allen1983] has been first introduced in the musical domain by Alan Marsden [Marsden2000] and their limits has been shown for musical analysis. Nevertheless, they are supposed to be sufficient for musical composition [Beurivé and Desainte-Catherine2001] and more convenient than points relations, because musical objects have generally a duration. Thus we propose a formalization of an interactive score, which is a static score augmented with temporal relations binding any of its parts [Desainte-Catherine and Brousse2003], and the definition of interactive points, that is,

starting or ending dates of either parts or notes that are distinguished. Then, an interpretation of such an interactive score consists in activating dynamically all interactive points during performance, while the system assures the consistency of the temporal relations of the resulting piece. Interpretation is thus limited to timing of the score events. Other events are not taken into account in the system presented in this paper.

The interest of this study is twofold. Firstly, interpretation of a written piece can be formally defined by the composer as a set of eventual pieces resulting from the temporal constraints that bind musical events. In that way, the composer can describe a kind of degree of freedom given to the musician, while he gets the certainty that temporal relations he specified between musical events will always be satisfied. Secondly, the same piece can be interpreted in several ways, by varying interaction points and temporal relations by a simple edition of the interactive piece. Thus, the piece could be adapted to any situation in a very simple way, according to the material and musicians that are available for performance.

We base our approach of musical interactivity on the work of Jean Haury about the musical interpretation [Haury, Haury and Schmutz1998]. He showed that the interpretation of a musical piece is possible through *interaction points* that are the trigger and the release of the notes and also with parameters of each note such as its magnitude. The interpretation for the player consists in choosing the date of execution of the *interaction points* and values for the other parameters. In the case of instrumental music, the composer can define and limit the possibilities of interpretation with annotations on the score such as magnitude indications (piano, forte, crescendo...) or tempo variations (accelerando, rubato...). In our study, we limit ourselves to what Jean Haury calls the "agogic modifications" that is the fluctuations of speed that temporally change the tempo. Precisely, these modifications are possible thanks to the *interaction points* : the player can accelerate or slow down the tempo through the dates of the trigger and the release of the notes.

Finaly our goal is to provide a model for creation and execution of musical pieces. In this model the specification of a piece must be close to musical language ; for the moment we focus only on temporal aspects and discrete controls, therefore we want the machine for execution to be generic and not specific to each piece.

2 Representation of scores

In this section, we define formally the data structures that are necessary to present the model. We call *interactive score*, a score melting static events and interactive events that are bound by a set of properties.

2.1 Static Score

We consider that the static score is structured in a temporal hierarchy [Beurivé and Desainte-Catherine2001]. In this paper, we focus on the temporal organization of the score, since sound representation is another subject and it is not investigated here.

Let us consider that a static score is a set of notes. Each note of the score is represented by a tuple: $n = < s, e, d, a, l >$, where s is the starting date of the node n, e is the ending date, d is the duration, a is a set of musical attributes and l is the list of children of n. When l is empty, the node is a leaf. Let us point out that in this paper, a simple leave is called a note, as well as a note containing other notes. Let us moreover define two functions for the notes : s and e, such that :

for $n = < s, e, d, a, l >$, $s(n) = s$ and $e(n) =$ e

We consider that the static score is completely specified. That is, every parameter has been given a value, so that the score can be played without any interaction.

Let us now present an example of such a score in Figure 1. In this example, the score will be defined as follows :

$P = < 0, 25, 25, a_P, (A, B, G) >$ where:

- $A = < 1, 7, 6, a_A, () >$
- $B = < 9, 24, 15, a_B, (C, D, E, F) >$ where $C = < 1, 6, 5, a_C, () >$, $D = < 6, 13, 7, a_D, () >$, $E = < 6, 8, 2, a_E, () >$, $F = < 7, 12, 5, a_F, () >$
- $G = <12, 19, a_G, () >$

2.2 Events

Two events are associated to a note, which are the starting and the ending events. In a static score, all events are considered to be static. They are launched automatically by the computer.

Definition 1. *Let s be a static score, an event is a couple $\epsilon = <n, t>$ where :*

- *n is a note of s;*
- *$t \in$ {start, end} is the type of the event;*

To define an interactive score, a set of events has to be chosen to admit interactive points. Such events are considered to be interactive and associated to a discrete control coming from a peripherals, and will be executed in real time during performance by a musician.

Definition 2. *Let s be a static score, an interactive point on s is a couple $i = <e, c>$ where :*

- *e is an event of s*
- *$c \in C$ is a discrete control from a control set C given by the entry peripherals.*

Definition 3. *Let s be a static score, an interactive event is an event e of s, such that their exists an interactive point $i = <e, c>$.*

2.3 Interactive Score

Let us define at this point of the paper what is exactly an interactive score. For temporal specifications, it involves elements that are introduced in the next section.

Definition 4. *An interactive score is defined by a tuple $m = <s, i, r, p>$ where*

- *s is a static score*
- *i is a set of interactive events*
- *r is a set of temporal relations $r=(a, n_1, n_2)$ as $a \in A$, $n_1 \in m$, $n_2 \in m$ where A is the set of Allen's relations.*
- *p is a set of tuples $<\Delta, \sigma, \rho>$ where Δ is a time interval occurring in s, σ is a status (breakable or unbreakable) and ρ is a priority (high or low).*

We shall denote by $\Sigma(m)$ the set of events of m.

2.4 Event Graph

An event graph is an directed graph, with a source and weighted arcs, formally :

$$G = < V, A, f, \mathrm{b}>$$

where V is the set of vertices, $A \subset V \times V$, f is a function labelling each arc by its duration: $f : A \longrightarrow N$, $b \in V$ is the source. For an interactive score s, we define an associated event graph such that $V = \Sigma(s)$, where $\Sigma(s)$ is the set of events of s.

3 Specification of Temporal Properties

Since the composer can introduce interaction points into his score, the dates of some events that are written in the static score will be no more respected. In order to prevent the piece from being totally destructured and to define more precisely the freedom of the musician during performance, it is necessary to specify temporal relations between the notes, relations that will be guaranteed during performance. These relations are the Allen's relation [Allen1983] that are presented in the first subsection.

Allen's relations provide a way to specify a partial order of appearance of the events during performance. If the values of durations and time intervals that are written into the score are forgotten, then, all interactive events will have a high priority, and will be able to reduce any time interval. We suppose that

this kind of mechanism would not always correspond to the will of the composer. As a matter of fact, any time interval separating an event e_1 from another event e_2, e_1 and e_2 being either starting or ending events of the same note or different notes, is considered from a musicological point of view [Forte1980] as a duration. Thus, every time interval may be involved in the rhythmic structure of the musical piece, as a basic element, and the composer should decide of the importance of each of them. For that purpose, we propose a generalization of the temporal properties of Allen, leading to a more powerful temporal specification. In order to express those temporal properties, we need to differentiate two kinds of time functions

- The absolute date function t of events that are written in the score;
- The absolute date t' of events during performance;

3.1 Temporal Relations between Notes

Allen's relations are represented on figure 2. They are defined between the notes and not between the events. This definition seems more convenient from our point of view for writing instrumental music using notes than a definition based on events which would be closer to a programming approach. Nevertheless, Allen's relations are only qualitative, while all temporal positions and durations are completely specified in the static score. Thus, we propose to keep this information and use it for expressing quantitative temporal properties in addition to Allen's relations (see next subsection). For that purpose, it is necessary to translate every Allen's relation. The figure 3 presents the transformation of each relation into an event graph. In such a graph, an arc (ϵ_1,ϵ_2) means that ϵ_1 must appear before ϵ_2 during the performance. Every time interval is specified and associated to an arc on the event graph. We can see on this figure that the duration of a note implies that its start must appear before its end, and the arc is labeled by the duration of the note.

3.2 Priorities of time intervals

The definition of interactive points implies that it is not possible to ensure every time interval of the static score during performance. In particular, an interactive event controls completely the time interval preceding its occurrence. It is part of the interpretation. As a consequence, in the event graph, the value of each time interval labeling an arc (ϵ_1,ϵ_2) where ϵ_2 is interactive, can be forgotten, because it is impossible to ensure any temporal property about it. So, this label will be replaced in the following by the discrete control which is associated to the event.

In a more general way, the apparition date of an interactive event during performance is different from the written date. As a consequence, dates and time intervals around the interactive events must be modified in order to verify the temporal relations if the interactive event appears too soon or too late. It is thus necessary to find another solution by computing other values for the time intervals and dates variables. For that purpose, it is important to give the composer the possibility of limiting the space of solutions, because it is not up to the system to decide what interval is more important than the other.

We have limited our study to the modification of the time intervals preceding or following an interactive event. In each case, we introduce two levels of priority, saying how the time interval can be modified. In that way, by defining all intervals to be of low priority, we go back to the mechanism which is only specified by Allen's relations.

In the case where the time interval precedes the interactive event, its priority says whether or not it can be interrupted if the interactive events appears too soon. Such a situation will be called a *backward reduction*. In the case where the time interval follows the interactive event, its priority says whether or not it can be reduced if the interactive events appears too late. Such a situation will be called a *forward reduction*.

3.3 Backward Reduction

Let us take as an example, the score which is represented on figure 4. This is a very simple example of two notes A and B bounded by a before relation, the start of B is an interactive event triggered by the control x and the other events are static. Time interval Δ can be forgotten as it precedes directly an interactive event. Time interval Δ_1 (resp. Δ_2) represents the duration of A (resp. B). The order of the events must be respected during the performance and $s(A)$ must appear before $e(A)$, as usual,

which must appear before $s(B)$. But during the performance, the player can trigger $s(B)$ before the time interval Δ_1 has passed and $e(A)$ has appeared. In this situation, we have two possibilities.

Firstly, to respect the order of the events, the computer triggers $e(A)$ before the end of Δ_1 and Δ_1 is be reduced. This situation is obtained by attributing the Δ_1 interval a low priority. In this case, we state the following property :

$$t'(s(B)) \geq t'(s(A))$$

Secondly, the interactive event $s(B)$ is not taken into account before the time interval Δ_1 has passed. This situation is obtain by attributing the Δ_1 interval a high priority. In this case, we state the following property :

$$t'(s(B)) \geq t'(s(A)) + \Delta_1$$

3.4 Forward Reduction

Some Allen's relations can be generalized in order to specify properties between quantitative information and interaction. We present in this paragraph the before relation which is represented on figure 5, which provides a case of forward time interval reduction.

In this example, the before relation holds between the notes A and B and the e_A event is interactive, so that the duration of A is forgotten. Let us study what properties can be stated concerning the time intervals Δ_1 and Δ_2. In this situation, if the player waits a time longer than $t(e(A)) - t(s(A)) + \Delta_2$ before triggering the interactive event $e(A)$, two kinds of situations may occur.

Firstly, this operation can shift the static event $s(B)$ in order to strictly respect the time interval Δ_2.

Secondly, by a strict respect of the before relation of Allen's, the quantity may be omitted and one can only consider an order between the dates. Thus, in that case, when the date of the end of A is shifted, the time interval between the end of A and the start of B should be reduced, dwindle to zero and consequently, $e(A)$ and $s(B)$ would be glued and $e(A)$ should shift $s(B)$.

By associating priorities to the Δ_1 and Δ_2 quantities, the composer is provided a way to choose between the two behaviors previously described. In fact, there are four possibilities:

- Δ_1 and Δ_2 have a high priority : if $t'(s_P) + \Delta_1 \leq t'(e_A) + \Delta_2$ then $t'(s_B) = t'(e_A) + \Delta_2$ else $t'(s_B) = t'(s_P) + \Delta_1$. This is the first situation presented.
- Only Δ_1 has a high priority : if $t'(e_A) \leq t'(s_P) + \Delta_1$ then $t'(s_B) = t'(s_P) + \Delta_1$ else $t'(s_B) = t'(e_A)$. This is the Allen's before behavior.
- Only Δ_2 has a high priority : if $t'(s_P) \leq t'(e_A) + \Delta_2$ then $t'(s_B) = t'(e_A) + \Delta_2$ else $t'(s_B) = t'(s_P)$.
- Δ_1 and Δ_2 have low priority : if $t'(s_P) \leq t'(e_A)$ then $t'(s_B) = t'(e_A)$, else $t'(s_B) = t'(s_P)$.

We can now generalize these rules for a configuration involving two time intervals as presented on figure 6. The four possibilities of priority can be stated as follows :

$$t'(C) = Max(t'(A) + k_1\Delta_1, t'(B) + k_2\Delta_2)$$

with $k_i = 1$ if Δ_i has a high priority, and $k_i = 0$ if Δ_i has a low priority.

3.5 Conflicts between Backward and Forward Reductions

In the case where the same intervals are involved in a backward and a forward reduction, it is necessary to differentiate the two priorities, because they are both needed to compute the resulting dates of the events. Thus, let us introduce the following terminology for backward reduction. In the case of forward reduction, we shall keep the terminology of high or low priority.

Definition 5. *A time interval is said to be breakable if it has a low priority for backward reduction. By contrast, it is said to be unbreakable if it admits a high priority for backward reduction.*

Let us consider a first simple example involving a configuration with two breakable time intervals which is represented on figure 7. In this example, the relations overlaps and before imply that the interactive event $s(C)$ might reduce the time intervals Δ_1 and Δ_2. Let us suppose that these two time intervals are breakable. We have got the same four possibilities than in the previous paragraph for the priorities :

- Δ_1 and Δ_2 have a high priority :

$$t'(e(A)) = Min(Max(t'(s(A)) + \Delta_1, t'(s(B)) + \Delta_2), t'(s(C)))$$

- Δ_1 has a high priority : If $t'(B) \leq t'(A) + \Delta_1$,

$$t'(e(A)) = Min(t'(s(A)) + \Delta_1, t'(s(C)))$$

  else

$$t'(e(A)) = t'(s(B))$$

- Δ_2 has a high priority : If $t'(A) \leq t'(B) + \Delta_2$,

$$t'(e(A)) = Min(t'(s(B)) + \Delta_2, t'(s(C)))$$

  else

$$t'(e(A)) = t'(s(A))$$

- Δ_1 and Δ_2 have a low priority :

$$t'(e(A)) = Max(t'(s(A)), t'(s(B)))$$

All these equations are available when $s(A)$ and $s(B)$ have appeared. In order to clarify the situation we present on figure 8 a temporal diagram for this example where both time intervals have a high priority. Figure 9 presents the same diagram for the case where only the time interval Δ_1 has a high priority.

To generalize this example, we have to study the priority rules between a breakable and an unbreakable time interval. But, this case is easier than the preceding one because there is only one possibility to describe, that is, the case where the two time intervals have a high priority. Indeed, the case where only the unbreakable time interval has a high priority is similar to the configuration of two unbreakable time intervals with one high priority and the case where the breakable time interval has priority is similar to the configuration of two breakable time intervals with one high priority. The case with no high priority is similar to the preceding no high priority cases. To illustrate the configuration of a breakable time interval and an unbreakable one, we can re-use the configuration of the figure 7 in which we consider that Δ_1 is unbreakable. With high priorities for Δ_1 and Δ_2, we have got the following equation:

$$t'(e(A)) = Max(t'(s(A)) + \Delta_1, Min(t'(s(B)) + \Delta_2, t'(s(C))))$$

This equation is available only when s(A) and s(B) have appeared. The figure 10 presents a temporal diagram to explain this configuration. A generalization of this formula is expressed in the following section with the formalism of the event three-colored graph.

4 Architecture of the system

The interactive score is translated into a simple competitor model according to the constraints that have been defined. Firstly, we build a three-colored graph, providing the partial order between all the events. Colors are used to indicate status of time intervals regarding interactivity. The three-colored graph is compiled into a static musical environment that will be used to execute an abstract machine reading its input from the peripherals and outputting synthesized sound. Figure 11 presents a global scheme of our system.

4.1 Three-colored Graph

A three-colored graph is an event graph whose arcs admit a color, formally :

$$G = < V,\ A\ , f_A,\ p_A,\ b >$$

where V is the set of vertices, $A \subset V \times V$, $f_A : A \to \{RED, BLACK, BLUE\} \times N$, $p_A : A \to \{0, 1\}$ and $b \in V$ is the source. For an interactive score s, we define an associated three-colored graph such that :

- $V = \Sigma(s)$
- $\exists\ a \in A, a = (\epsilon_o,\ \epsilon_t)$ only if $t'(\epsilon_o) \leq t'(\epsilon_t)$
- The color of an arc represents the backward reduction priority of the time interval, as well as the property of been just before an interactive event, as follows : For $a = (\epsilon_o, \epsilon_t) \in A$ and $\Delta \in N$:
  - $f_A(a) = (BLUE, \Delta)$ means : $t'(\epsilon_t) \geq t'(\epsilon_o) + \Delta$
  - $f_A(a) = (BLACK, \Delta)$ means : $t'(\epsilon_t) \geq t'(\epsilon_o)$
  - $f_A(a) = (RED, \Delta)$ means : ϵ_t is interactive
- The forward reduction priority of an arc is represented by the function p_A, 0 representing a low priority and 1 a high priority.

4.1.1 Red Arcs

An arc is RED if its source event precedes its target event, and this target event is interactive. Such an arc does not carry any duration value. Thus, it is possible to apply simplifications while building the graph from the interactive score.

4.1.2 Black and Blue Arcs

An arc is BLUE if, during performance, the duration between its source and its target events has to be greater or equal than the one written in the score. It corresponds to the unbreakable time intervals that have been introduced in section 3. In this case, the composer gives a higher priority to the written duration than to interactivity.

On the contrary an arc is colored in BLACK when the composer allows to interrupt its duration because of interaction. It corresponds to the breakable time intervals that have been introduced in section 3.

Forward reduction priorities that have been defined in section 3 are now associated to blue and black arcs by the means of the p_A function. Let e_d be a static event and let $\forall i \in [1, n]$, (a_i, e_d) is blue, and $\forall i \in [1, m]$, (b_i, e_d) is black. Let us define J_n such that $\forall i \in J_n$, $p_A(a_i, e_d) = 1$, and J_m such that $\forall i \in J_m$, $p_A(b_i, e_d) = 1$. Let us define the following quantities

- $M_n = max(t'(A_i) + \delta_{a_i}))$ for $i \in J_n$
- $M_m = max(t'(B_i) + \delta_{b_i}))$ for $i \in J_m$
- $M_l = max(t'(A_i) + t'(B_j))$ for $i \in J_n$ and $j \in J_m$

Then, if $M_l \leq max(M_n, M_m)$, then $t'(C) = max(M_n, min(M_m, t'(D)))$ else $t'(C) = M_l$.

4.2 Petri Network

Petri networks are a general-purpose tool for handling concurrency [Murata1989]. They have been used in the computer music field by several authors. One of the main studies has been conducted by Goffredo Haus [Haus1994] in the domain of formal representation of scores, and structural descriptions of music that are suitable for communication of music information among composers, musicologists, scientists and listeners. Music objects are then associated to transformation processus that are described by Petri networks. These studies are based on an analysis of musical pieces and of the compositional process that lead to them. Thus, our purpose is very different since we propose a model that convey information from the composer himself to the performer. Moreover, by contrast with these previous studies based

on rhythmic, melodic and harmonic objects, we focus on very basic timing objects like activations and release of notes.

The system is based on a set of partially ordered events. Nevertheless, at execution time, real-time events will admit a total order. Because of the indeterminism of this order (several orders can eventually occur at performance), finite automata is not a suitable representation, even indeterministic automata. As a matter of fact, the specification of all possible cases of orders occurring between events would be necessary, leading to a high number of states. On the contrary, Petri network are well suited for handling concurrency.

4.2.1 Definition

A Petri Network is an directed graph with two types of vertices, the places and the transitions: $PN = < V, A >$, with $V = P \cup T$, where P is the set of places and T, the set of transitions, and $A = (P \times T) \cup (T \times P)$.

Each place contains a number of tokens greater or equal to zero. Every transition contains a condition which have to be satisfied for tokens to cross it. Moreover, all places admitting an arc towards a transition t have to contain at least a token for the transition t to be passed. When a transition t is passed, one token is removed from all places preceding t and one token is added to all places admitting an arc coming from the transition t. Then, execution of a Petri network is a sequence of tokens moves.

In our case, the source of the graph is the beginning of the musical piece. Initially, one token is given to the source. Transitions conditions provide a way to wait for input controls (in the case when an interactive event is expected) and for a time interval (in the case when a written time interval has to be respected). In addition, actions are associated to places and are used to launch events. Only two types of functions are necessary, functions $ON(n)$ (to trigger the note n) and $OFF(n)$ (to release the note n).

4.2.2 Places

We first made the hypothesis that it was not necessary to associate a Petri network place to all events of the score, but only to interactive events ([Dessainte-Catherine and Allombert2004]). This hypothesis was based on the idea that the dates of launching static events could be computed and stored in a queue, waiting for execution. As a matter of fact, it appeared that because of the problem of backward and forward reduction of the duration of black arcs, it was not always possible to compute all the launching dates of the static events. Thus, it appears to be necessary to separate events, by putting only one event in each place (except for those which have exactly the same launching date). As a consequence, only one action is stored in a place, which depends on the nature of the event: $ON(n)$ (to trigger the note n) and $OFF(n)$ (to release the note n).

4.2.3 Transitions

Each transition admits a logical formula. This one has to be satisfied for the transition to be crossed. A logical formula is a disjunction of predicates of the two following kinds.

- Wait(X) is a predicate that permits to wait for the duration X. Let h be the value of the clock at the moment of the first call of the predicate. The value of each call to Wait(X) is false while $h + X$ is less than the current value of the clock, and becomes true when $h + X$ is greater or equal than the current value of the clock.
- Get(X) is true if the event X has arrived in input.

4.3 Construction of the Petri network

The figures 12 and 13 present the transformation of two configurations of three-colored graph into Petri network. The first example of the figure 12 only involves blue and red arcs, since the weight of the blue arcs must be respected, they imply transition admitting Wait predicate. The interactive event implies a transition admitting a Get predicate. One can easily verify that the temporal properties expressed by the graph are maintained by the Petri network.

The figure 13 presents the Petri network coming from the transformation of a black arc and give and example of a transition admitting a logical formula. As it is described in subsection 3.5, the temporal formula for computing the launching time of B event is the following:

$$t'(B) = Min(t'(A) + \Delta, t'(C))$$

Once again, one can easily verify that the Petri network ensures this property.

4.4 Execution of the Petri network

We describe the evolution of such a Petri network in the figure 14. At the beginning, there is one token in the place of A. Since the transition following the place of A admits no predicate, it's crossed immediately. From this situation, there are two possibilities ; if the control x is activated before the end of Δ, the predicate "$Wait(\Delta)$ *or* $Get(x)$" is true so a token is created in the place of C, and the predicate "$Get(x)$" is also true, so a token is created in the place of B. If the control x is not activated before the end of Δ, the predicate "$Wait(\Delta)$ *or* $Get(x)$" becomes true and a token is created in the place of B, but not in the place of C. Consequently, during the performance after A appears, if the player activates the control x before the end of Δ B and C will be triggered at the same time, so the order of the events is respected but the delay Δ is reduced. On the contrary, if the player doesn't activate the control before the end of Δ, the event A will be triggered at the date $t'(A) + \Delta$ and C will be triggered when the player activates x. Finally, this Petri Network permit the respect of the behavior of a black arc.

5 ECO Machine

Abstract machines are largely used in the domain of programing languages, especially in functional language interpretation (lisp language) and provide a way to specify formally the execution of a program depending on its instructions and its environment.

5.1 Definition

An ECO machine is an abstract machine such that:

- a state of the ECO machine is a 4-tuple (E, C, O, t) where:
  - E is a musical environment, that is represented by the Petri network;
  - C is a control string representing input time-stamped events;
  - O is the output string;
  - t is the time-stamp of the state.
- the operation of the machine is described in terms of state transitions that are synchronized on a clock. The first state is associated to the initial date 0. Let δt be the value of a cycle of the clock, transitions occur at that rate. Given the current state (E, C, O, t), the next state $(E', C', O', t + \delta t)$ is determined by the events of the current control string C whose time-stamp are greater than t and lower than $t + \delta t$.

5.2 Execution

Concretely, the execution consists in activating all the places of the Petri Network that contain a token. The Petri Network begins with a source which is a transition. This transition is managed by the signal for the start of the piece which is always interactive. Then, at each step of the loop, the following operations occur.

1. Watch input events from the control string C and maintain the table of events;
2. Activate each place of the Petri network E by executing its action and send the result to the output O;
3. Cross each transition of E that satisfies the conditions, current time being the value of t;

The performance stops the loop when the end of the piece has been run.

6 Applications

In the model of interactive scores, any event of a score can eventually be an interactive point, that is, it maybe controlled by a peripheral, for example a button or a key, which are often used as discrete controls. From the point of view of the musician, every interactive point is associated to an *action*, that is, a gesture activating the corresponding peripheral.

Let us now overuse this term and consider that every event of a score is associated to an *action*. If the event is an interactive point, then the action is a gesture from the musician, while if the event is static, the action is executed by the computer. In other words, let $A(s)$ be the set of all actions that are associated to a score s. An interactive score, $m =< s, i, r, p >$, splits the set $A(s)$ into two sets, i containing actions for the musician (interactive events) and $c = A(s) \setminus i$ containing actions for the computer (static events). In the following, we shall use the notion of *level of interaction for events* of an interactive score m, which is defined by the formula $I(m) = \frac{|i|}{|A(s)|}$.

In the first subsection, we present two examples of simulation of interaction existing in real-life, in order to give a concrete idea of the potential of the model and the measure on interaction. Of course, simulating such existing interactions is necessary to design a model. Nevertheless, generalization of those interactions permits the specification of new kinds of interactions that provide interesting applications. Next subsections present two applications of interactive score providing new kinds of interactions that we can call *augmented play*, like augmented reality in the domain of virtual reality.

6.1 Real-life Musical Interactions

Among the various forms of interaction that take place with actual instruments, we find for example, the orchestra conduction. For simulating such an interaction with the computer, it is necessary to specify the whole score to play as well as the instruments composing the orchestra. Then, the conductor can control the volumes, the mixing, the speed and the beginnings and endings of the principal parts of the score. Especially when the orchestra is big, most of the musical events are not directly controlled by the conductor, but only by musicians. The level of interactivity of the orchestra conductor is strictly less than 1.

Instrumental interaction that takes place between a musician and his acoustic instrument is more complex since the former has to control every beginning and ending of each note, as well as every sound parameter provided by its instrument. Simulating such an interaction with the computer does not necessitate to specify the score at all, since the musician controls all the events in real time. The level of such an interaction is equal to 1.

6.2 Gesture-assisted Interaction

The model of interactive score provides a way to adapt any score to the ability of any musician, from a young child or a disable person up to a virtuoso. The more the musician is a virtuoso, the more the level of interaction of the score can be close 1.

6.3 Pedagogical Application

One of the areas of application is music education. Whatever is the studied instrument, the learner has to begin playing very simple musical pieces to be able to play them. This is due to the fact that he has to control every parameter of the produced music: activation and releasing, as well as pitch, volume, modulation and timbral aspects of all the notes appearing in the score. The system resulting from the study presented in this paper, should provide a way to simplify the interaction of the learner without loosing musical interest, so that the learner could enjoy playing a large set of pieces, even very difficult ones. Interaction level should augment while the learner makes progress.

6.4 The Orchestra Against Silence

The *Orchestra Against Silence* [Haury and Schmutz1998] is composed of disable persons. The orchestra uses special sensors adapted to the ability of each musician, and plays with the assistance of the computer. The computer is used to assist the gesture of the musicians, by automatically executing actions that are specified in the score. In particular, pitches of the notes are fixed into the score. In most of the cases,

beginnings of notes are executed by the musicians, and the velocity of their gesture is transmitted as a parameter of interpretation.

Let us take the score of figure 15(a) as an example, and let us focus on the temporal structure, putting aside the velocity parameter. The seven notes of the right hand will be called A, B, C, D, E, F and G and the two notes of the left hand H and I. The interactive score which is represented on figure 15(b) specifies that the beginnings of the notes of the right hand are all interactive points (represented by a pattern), while other events (endings of the same notes and beginnings and endings of notes H and I) are static. Moreover, the *meet* temporal relations, which is denoted by m on the figure, specifies that the corresponding notes compose a melody, and the *start* relations, which are denoted by s specify the synchronization of notes D and H, as well as F and I.

This interactive score is an example for one nusician. If two musicians are available to play this score, one can associate interactive points to beginnings of notes H and I and remove the relations *start*.

6.5 Artistic Applications

An other artistic interest of this model is that a composer can create different versions of an interactive piece based on the same static score simply by changing the points of interaction. Several aspects of the performance context can be taken into account. For example, an acousmatic version of an interactive piece involving one or more musicians can be easily obtained by removing all points of interaction, and vice-versa. Moreover, some sensors problems can be solved by removing corresponding interaction points or changing interaction association. Even the mood of the composer or the performers can also provide various versions of the piece.

7 Conclusion

We presented a formal definition of interpretation of written piece as a set of pieces obtained during performance and satisfying certain temporal constraints between triggering and releasing events. We proposed an operational model to compute a program associated to any kind of interpretation associated to a musical piece. A system implementing this model is under development. This system should provide a way to define interaction points on a score bound with temporal relations and to associate them to any kind of peripherals providing discrete control (keyboard, mouse, etc.).

Next step of this research consists in generalizing the model in order to expand the influence of an interactive event to a larger time interval. In this paper, we only consider two time intervals that can be affected by real-time interaction, those that precede or follow an interactive event. A generalization should provide a model in which we are able to suppress static events according to what happens in real-time. Then, we would like to study durations constraints in order to specify rhythmic controls in real-time. At last, of course, it will be necessary to integrate continuous controls in the model. This should not change fundamentally the model, unless mapping has to be defined between continuous control and musical or sound parameters. For that purpose, the hierarchical structuring of the score will be very useful for structuring the mapping itself. As a matter of fact, hierarchical mapping will result from this model and will provide a very comfortable way to represent interactions with sound synthesis as well as interactions with musical parameters.

Acknowledgment

We would like to thank Jean Haury and György Kurtag for sharing with us their very precious experience with interpretation and improvisation.

This research was carried out in the context of the SCRIME[1] project which is funded by the DMDTS of the French Culture Ministry, the Aquitaine Regional Council, the General Council of the Gironde Department and IDDAC of the Gironde Department. SCRIME project is the result of a cooperation convention between the Conservatoire National de Région of Bordeaux, ENSEIRB (school of electronic and computer scientist engineers) and the University of Sciences of Bordeaux. It is composed of electroacoustic music composers and scientific researchers. It is managed by the LaBRI (laboratory of research in computer science of Bordeaux). Its main missions are research and creation, diffusion and pedagogy thus extending its influence.

[1] Studio de Création et de Recherche en Informatique et Musique électroacoustique, www.scrime.u-bordeaux.fr

References

[Allen1983] Allen, J. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

[Beurivé and Desainte-Catherine2001] Beurivé, A. and Desainte-Catherine, M. (2001). Representing musical hierarchies with constraints. In *Proceedings of CP'01, Musical Constraints Workshop*.

[Camurri et al.1995] Camurri, A., Catorcini, A., Innocenti, C., and Massari, A. (1995). Music and multimedia knowledge and reasoning: the harp system. *Computer Music Journal*, 19(2):34–58.

[Desainte-Catherine and Brousse2003] Desainte-Catherine, M. and Brousse, N. (2003). Towards a specification of musical interactive pieces. In *Proc. of the CIM XIX, Firenze, Italy*.

[Dessainte-Catherine and Allombert2004] Dessainte-Catherine, M. and Allombert, A. (October 2004). Specification of temporal relations betwenn interactive events. *Proc. of the SMC 2004 (Sound and Music Computing), Paris, France*.

[Forte1980] Forte, A. (1980). Aspect of rythme in webern's atonal music. *Theory Spectrum*, 2:90–109.

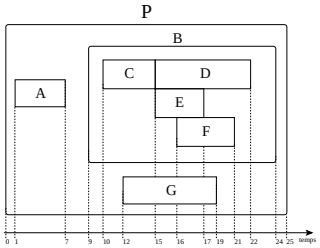
[Haury] Haury, J. La grammaire de l'exécution musicale au clavier et le mouvement des touches. In *Manuscrit*.

[Haury and Schmutz1998] Haury, J. and Schmutz, J. (1998). L'orchestre contre silence. In *Proc. of the JIM'98, Marseille, France*, pages D5–1.

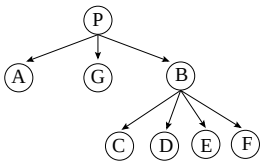
[Haus1994] Haus, G. (1994). *Music Processing*. Oxford University Press.

[Marsden2000] Marsden, A. (2000). *Representing Musical Time: A Temporal-Logic Approach*. Swets & Zeitlinger.

[Murata1989] Murata, T. (1989). Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77(4), pages 541–580.



(a) A graphical representation of P



(b) The tree representation of P

Figure 1: An example of a static score P

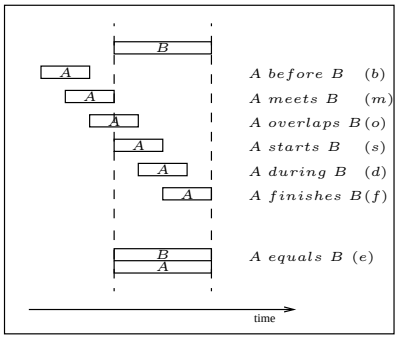


Figure 2: The Allen's relations

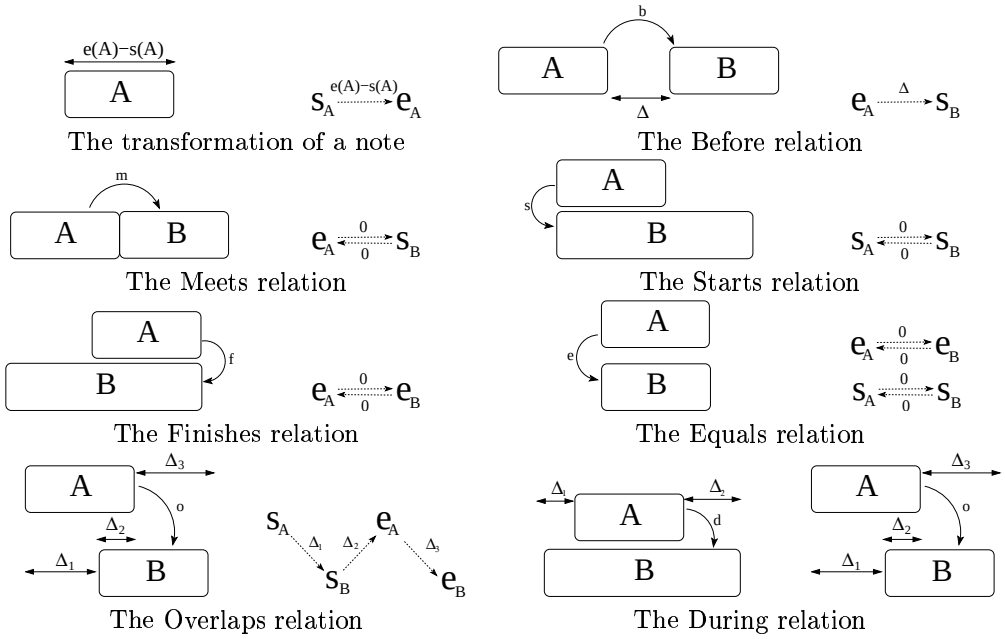
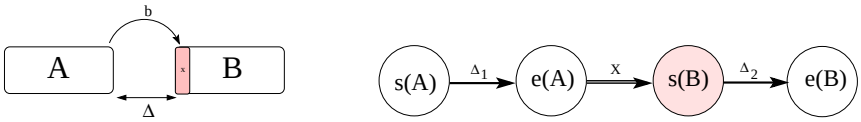


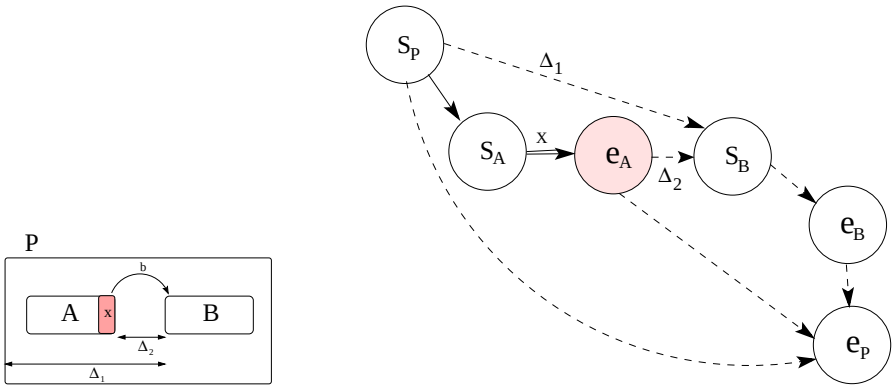
Figure 3: Correspondence between Allen's relations and event graphs



(a) Two notes related by a before relation

(b) The relations between the events

Figure 4: An example of backward reduction



(a) The notes related by a before relation

(b) The relations between the events

Figure 5: An example of forward reduction

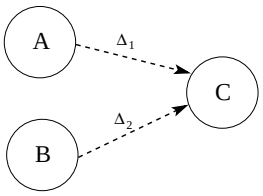
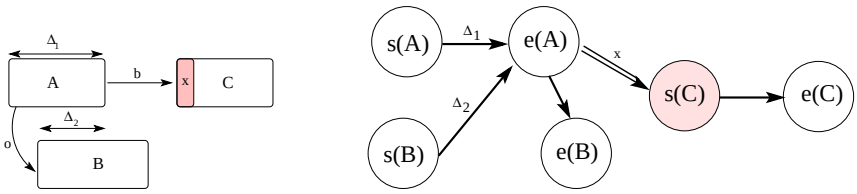


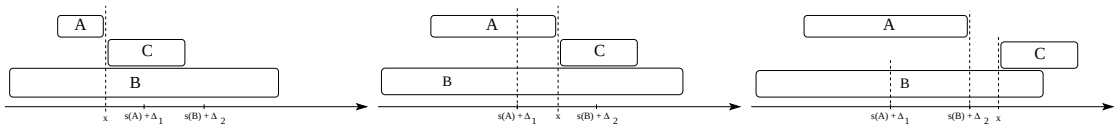
Figure 6: Generic case for forward reduction of two time intervals



(a) The notes configuration

(b) The events configuration

Figure 7: Forward an backward reduction involving two breakable time intervals

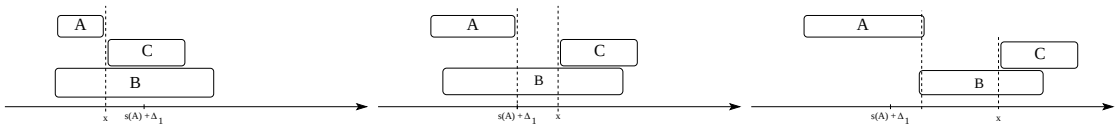


(a) Case where both time intervals are reduced

(b) Case where 1 time interval is reduced

(c) Case where no time interval is reduced

Figure 8: A temporal diagram representing the situation where both time intervals have a high level priority for forward reduction

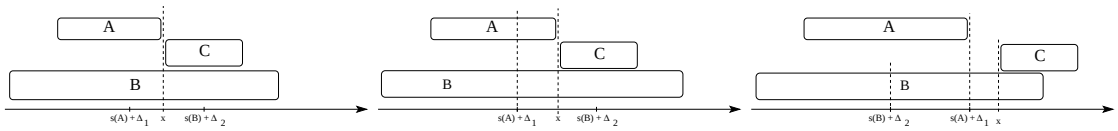


(a) Case where the time interval is reduced

(b) Case where the time interval is not reduced

(c) Case where we wait s(B) to trigger e(A)

Figure 9: A temporal diagram representing the situation where one time interval has a high level priority for forward reduction



(a) Case where the time interval Δ_2 is reduced

(b) Case where the time interval Δ_2 is not reduced

(c) Case where we wait Δ_1 to trigger e(A)

Figure 10: A temporal diagram to explain the situation where one time interval has priority

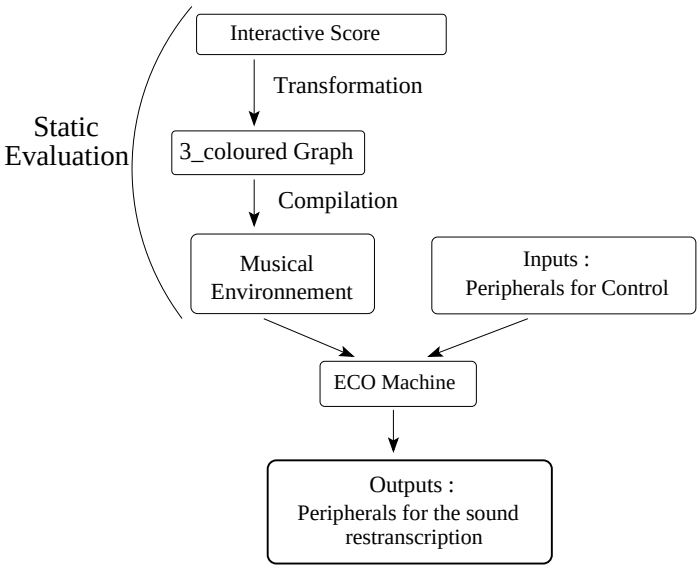
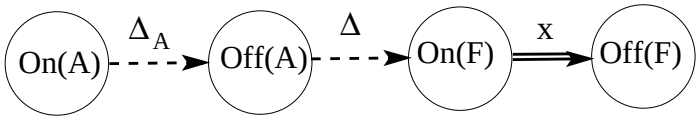
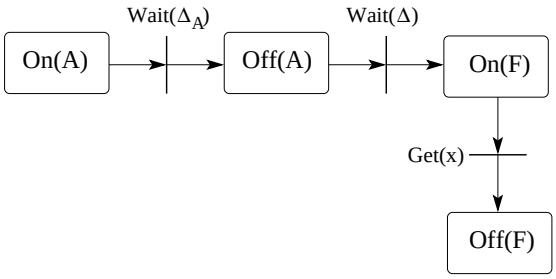


Figure 11: Architecture of the system

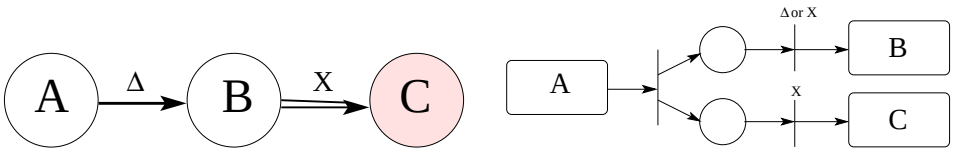


(a) Three-colored graph



(b) Petri net associated to the graph

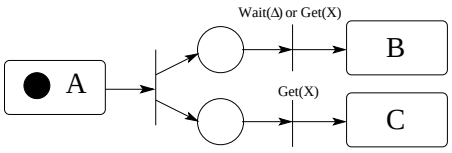
Figure 12: Transformation of a configuration of graph



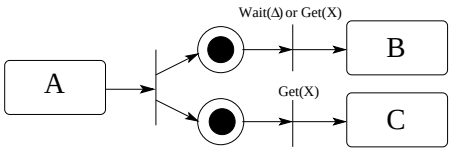
(a) Three-colored graph involving a black arc

(b) Petri net associated to the graph

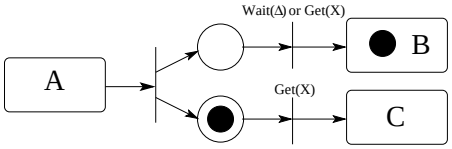
Figure 13: Transformation of a configuration involving a black arc



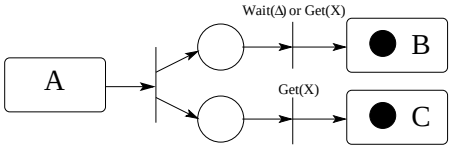
(a) Beginning : one token is the place of A.



(b) The transition following the place of A is crossed.

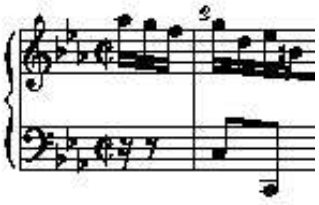


(c) Case where the control "x" is not activated before the end of Δ.

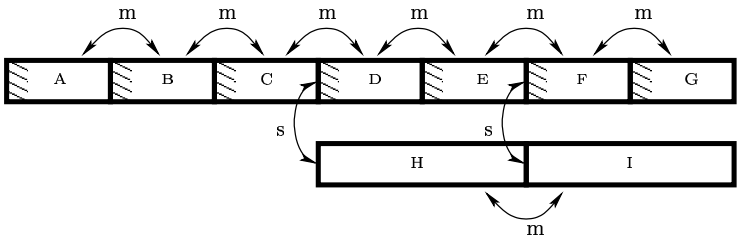


(d) Case where "x" is activated before the end of Δ.

Figure 14: The evolution of the Petri Network associated to a black arc



(a) A score for one musician



(b) Example of interactive score

Figure 15: The score on figure 15(a) can be associated to the interactive score of figure 15(b) where the musician controls only the beginnings of notes A, B, C, D, E, F and G. Other events (endings of the same notes and beginnings and endings of H and I) are launched by the computer according to the temporal relations that are specified (m for *meet* and s for *start*)