SPECIFICATION OF TEMPORAL RELATIONS BETWEEN INTERACTIVE 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@enseirb.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 contemporain 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 researchs 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 researchs 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 unifiy the different ways of interaction of musical and sound levels and bring a better understanding of musical interaction.

We start this study by focussing on temporal aspects of interaction, putting aside, for now, sound synthesis. 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 [All83] has been first introduced in the musical domain by Alan Marsden ([Mar02]) and their limits has been shown for musical analysis. Nevertheless, they are supposed to be sufficient for musical composition [BDC01]) 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 Allen's temporal relations binding any of its parts [DCB03, All83], 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.

Interpretation of musical pieces based on the operations of activating and releasing notes has been very well studied by Jean Haury [Hau, HS98]. In this research, the piece is entirely written, and the musician can activate musical events. He has the choice of the starting dates, velocity and ending dates of the events. Such a study showed how important are temporal relations between those dates for specifying linking between notes.

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. Another area 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 could get more and more difficult and follow the learner's progress.

2. ARCHITECTURE OF THE SYSTEM

The building of the interactive piece proceeds in several steps. Given a static score, some starting or ending dates can be declared interactive and a peripheral is thus associated to them. That means that they will be triggered in real time by this peripheral. We call the result of this processus *interactive score*. Then, temporal relations binding starting dates and ending dates of its components can be defined in order to constraint the interactive score. The interactive score is then translated into a simple competitor model according to the constraints that have been defined. Firstly, we build a three-coloured graph, providing the partial order between all the events. Colours are used to indicate priority between static and interactive events. The three-coloured graph is compiled into a static musical environment that will be used to execute an abstract machine reading its input from the peripherals and outputing synthesized sound.

2.1. Static Score

We consider that the static score is structured in a temporal hierarchy [BDC01]. In this paper, we focus on the temporal organisation of the score, since sound representation is another subject and it is not investigated here. Thus, let us consider that each node of the musical hierarchy is represented by a tuple: $n = < s, e, d, a, l >$, that will be called a note, 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 notice that a simple leave we call a note as well as a node containing other nodes. Let us moreover define two functions for the notes : s and e, such that :

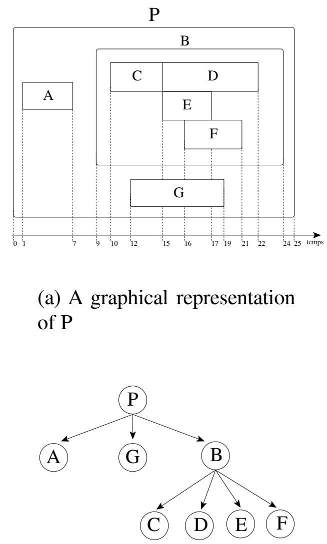
for $n = < s, e, d, a, l >$, $s(n) = s$ and $e(n)$ = n

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, $()$ >



(a) A graphical representation of P

(b) The tree representation of P

Figure 1. An example of a static score P

2.2. Temporal Relations

In order to produce an interactive score, the composer can introduce interactive points into his static score. This implies that the musician will chose the date of the start and end of some notes during performance. Thus, the respect of the dates that have been written in the static score is no more guaranteed. In order to define more precisely the freedom given to the musician at performance time, and how its interpretation can be different from the written score, we provide the composer a way to specify temporal relations between notes, relations that will be guaranteed during performance. These relations are the Allen's relations [All83] which are represented on figure 2. Since Allen's relation are only qualitative, we decided to introduce values (durations and delays) in relation with the static score as we can see in the Figure 4. Thanks to those values, constraints on durations and delays can be specified and interpretation can be closer to the score.

2.3. Interactive Score

Interactive score is built from a static score by choosing some interactive events and binding the notes with temporal relations. An interactive event is always associated to a starting or an ending date of a note. Thus, an interactive score is a tuple: $s = < m, i, r >$, where m is a static score, i is a set of either starting or ending dates of notes of m, that is $i = \{(n, t, c) | n \in m, t \in \{s, e\}, c \in C\}$, where s means *starting date* while e means *ending date* and c is a discrete con-

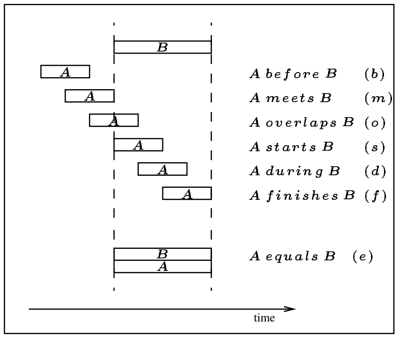
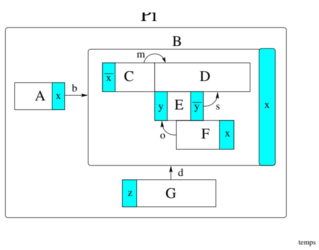
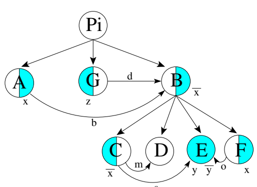


Figure 2. The Allen's relations



(a) A graphical representation of Pi



(b) The tree representation of Pi

Figure 3. An example of an interactive score Pi

trol provided by peripherals. At last, r is a set of temporal constraints binding notes of m, that is $r = \{(a, n_1, n_2) | a \in A, n_1 \in m, n_2 \in m\}$, where A is the set of temporal relations described in the preceding subsection.

Let us present in Figure 3 an example of an interactive score based on the static score P. In this example we have

Pi =$< P, i, r >$ where :

i = $\{(A, end, x)$, (B, end, x), $(C, start, \overline{x})$, $(E, start, y)$, $(E, end, \overline{y})$, (F, end, x), $(G, end, z)\}$

and

r = $\{(before, A, B)$, $(meets, C, D)$, $(starts, E, D)$, $(overlaps, F, E)$, $(during, G, B)\}$

We have to notice here that the hierarchical structure of the static score implies a *during* relation between a note and all of its children. These relations are not stored in the element t of the interactive score, but they will be used later.

As we can see, the main object of an interactive score is the *event*. An event ϵ of an interactive score s=$< m, i, t >$ can be either an interactive event, that means $\epsilon \in i$, either a static event that means ϵ = $(n, t) | n \in m, t \in \{s, e\}$. Let us denote by Σ_s the set of all the events of an interactive score s. We define an absolute date function t_a on Σ_s as: for $\epsilon \in \Sigma_s$, ϵ = (n, t, c) or (n, t)

$$t_a(\epsilon) = \begin{cases} s(n) & if\ t = start \\ e(n) & if\ t = end \end{cases}$$

The t_a function gives the date that is written in the static score. We cannot assume that during the performance, this will be the date of the event, because of the shiftings due to the interactive events. So we define another date function t_p which represents the effective date of the event during the performance. We cannot precisely express this date, but we can partially order these effective dates of the events thanks to the time relations between the notes. We now want to turn the interactive score into a representation that uses the events as main objects instead of the notes. This representation is the three-colored graph.

2.4. Three-colored Graph

A three-colored graph is an directed graph, with a source and colored and weighted arcs, formally :

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

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

$$V = \Sigma_s$$
$$\exists\ a \in A, a = (\epsilon_o, \epsilon_t) \text{ only if } t_p(\epsilon_o) \leq t_p(\epsilon_t)$$

The color of an arc is defined as follows :

For $a = (\epsilon_o, \epsilon_t) \in A$ and $\Delta \in N$:

- $f_A(a) = (BLUE, \Delta)$ means : $t_p(\epsilon_t) \geq t_p(\epsilon_o) + \Delta$
- $f_A(a) = (BLACK, \Delta)$ means : $t_p(\epsilon_t) \geq t_p(\epsilon_o)$
- $f_A(a) = (RED, \Delta)$ means : ϵ_t is interactive

An arc is colored in BLUE if, during execution, the duration between its source event and its target event has to be greater or equal than the one written in the score during execution. 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.

2.5. Musical Environment

Petri networks are a general-purpose tool for handling concurrency [Mur89]. They have been used in the computer music field by several authors. One of the main studies has been conducted by Goffredo Haus [Hau93] 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 rythmic, melodic and harmonic objects, we focus on very basic timing objects like activations and release of notes.

2.5.1. Petri Network

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 indeterminist 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. 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 specific dates (in the case when a written duration 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).

2.5.2. Priority Queue

Each place of a Petri network is associated to a set of actions. Some actions, that we shall call *immediate actions*, consist in launching an event immediately, while others consist in waiting for a certain date before launching the event. In a very concrete way, those actions consist in fact in adding in a priority queue a frame $< f, d, n >$ such that f is a pointer to the function (ON or OFF), d is the date for launching the function, and n is the note containing the event (start or end), providing attributes for applying the function.

2.5.3. State Variables

Indeterminism implies several possible total orders during performance between events. At static time, it is necessary to consider all those orders. That means that several interval of dates can be associated to the launching of certain events during performance especially when these events depend on several previous interactive events. The solution we adopted consists in adding in the queue all action frames for launching events for all possible cases. Then, in order to satisfy temporal constraints, we have to wait for the last action frame associated with an event e for really launching the event e.

For implementing this mechanism, we define a counter for each event e. This counter is initialized to the number of possibilities of launching that can occur at performance for the event e. Then, each time an action frame $< f, d, e >$ is removed from the priority queue, the counter of the event e is decremented. When the counter of an event e reaches zero, the function f is applied to the note n.

3. ALGORITHMS

3.1. Interactive Score Translation

We present here the steps that turn an interactive score into a musical environnement. We present two algorithms and discuss the conservation of all the information stored in the interactive score.

3.1.1. Three-colored graph

The first step consists in turning the interactive score into a three-colored graph which is a representation of the score based on the events. This algorithm is based on a set of elementary transformations that turn particular configurations of the interactive score into sub-three-colored graphs. We can find in this set the transformation of a note and the transformations that traduce the time relations between notes (see figure 4). For each case, we present the time relation in the interactive score with the associated three-colored graph. In these figures, we note an event as follows: $s_A = (A,$

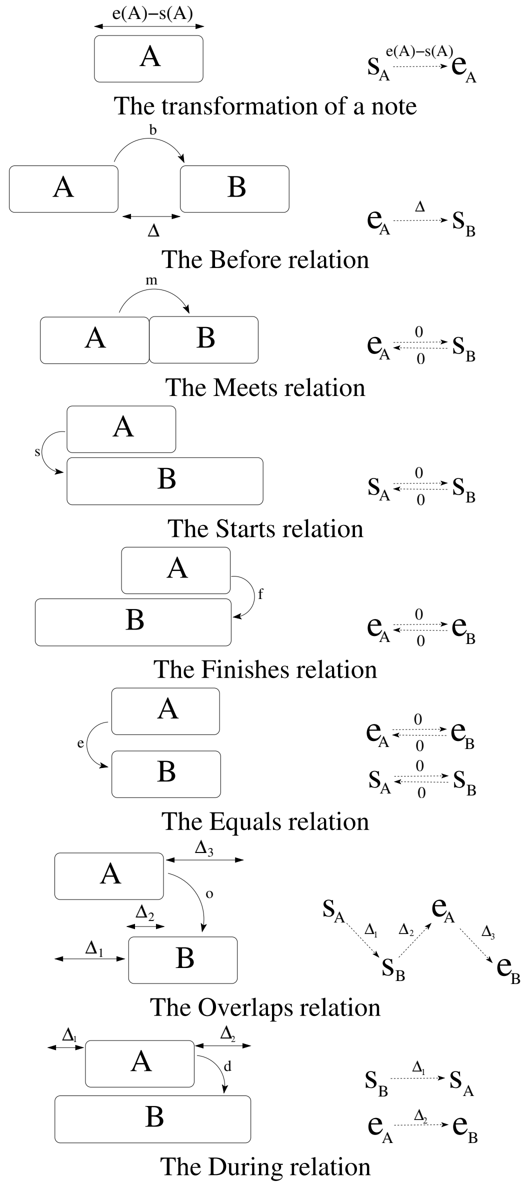


Figure 4. The elementary tranformations

$start$) and $e_A = (A, end)$, where A is a note. From now and for all the figures we will present, the following nomenclature is respected : a BLACK is drawn with a single solid line, a BLUE arc is drawn with a single dashed line and a RED arc is drawn with double solid arc.

We present the sub-three-colored graphs with generic events and arcs, without taking into account interactivity. So, in the figures, the events are static and the arcs are blue.

Once the elementary transformations are defined, the algorithm consists in creating a three-colored graph with a vertice for each event of the interactive score. It places arcs corresponding to the duration of the notes, and places arcs corresponding to the time relations be-

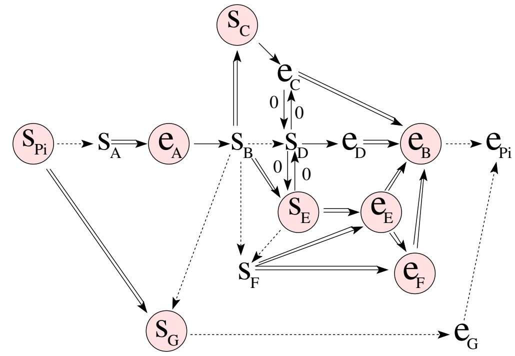


Figure 6. The associated three-colored graph of Pi

tween notes including the $during$ relations implied by the hierarchical structure. Each time a BLUE arc is preceded by a RED one, the composer is asked for the priority he wants: written duration or interactivity. If the composer chooses the duration, the BLUE arc is turned into a BLACK one, elsewhere the colors remain as they are.

We present in Figure 6 the associated three-colored graph of the interactive score Pi. We did some choices as a composer would do to determine the BLACK \ BLUE color of arcs followed by a RED arc. We don't present the weight of the arc (except 0), to not over complicate the figure. In reality, each arc is weighted except the RED ones.

3.1.2. Musical Environment

Now, we have to turn the three-colored graph into a musical environment that will be executed by the ECO Machine.

The first step consists in labeling events in the three-colored graph with places. Those labels indicate, for each event, which places contain its launching action. Since interactive events have only one launch action, they receive only one label. On the contrary, static events have as many launching actions as interactive events that preceed them in the partial order, so that launching of static events may be represented in several places. Thus, each interactive event is labeled by its own place, while static events are labeled by places of all interactive events preceding them with an BLUE or BLACK arc.

Once the events are labeled, we can create the Petri Network. We can find two types of places in it: the main places that contain the actions, and the empty places, which do not contain any action, and only permit appointments between events as well as the expectation of signals from peripherals. For each label l, we create a new main place, in which we store all the actions that correspond to the events of the three-colored graph that have been labeled with l. A main place t is directly reachable from a main place s, if there are arcs between the events corresponding to s and the events corresponding to t. As there is a mapping between interactive events and main places, we can see the Petri Network as the partial arrangement of groups of actions depended of an interactive event. As a consequence, empty places are created to represent all the conditions of the passing from a place to it successor, as well as immediate transitions (without any conditions) to satisfy alternance between places and transitions.

Let us illustrate this point with the situation between the end of note A and the the end of note F in the example of Pi. The Figure 7 shows the part of the Petri Network corresponding to this part of the three-colored graph. So, we find in the place of e_A (p_1), the instruction for s_B and s_F, because they are joined by BLACK and BLUE arcs. As we can see on the graph, the composer has chosen to give higher priority to the duration between s_B and s_F and a lower priority to the interactivity of e_F. So we must wait for s_F, before allowing the musician to end F. This constraint is satisfied thanks to the two empty places $p_duration$ and $p_peripheral$ and the transitions between them. Once s_F has been launched, we accepted the signal x for activating the place p_2.

Concerning other components of the musical environment, the queue is empty at the beginning of the execution, and the state variable of each event ϵ is initialized with the number of labels it has received during the labeling algorithm. As a matter of fact, it corresponds to the number of actions for launching ϵ that will be stored in the different places of the Petri Network.

3.2. ECO Machine

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, as it is described in the previous subsection;
  - 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 $t + \delta t$.

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. In general, when a main place is activated, the immediate actions it contains are consumed and the others are placed in the queue ordered by their date of completion. Each time an action is consumed, we decrement the state variable of the event that corresponds to this action. When a state variable is null, we effectively run the associated action, because it means that all the occurrences of the action have been consumed, so that all the constraints on this action have been satisfied. The performance stops the action when the end of the piece has been run.

3.2.0.1. Computational Complexity

Complexity of the system has not been yet studied precisely. Nevertheless, we can already say that it mostly depends on the number of actions that are associated to each place of the petri network. Each of those actions being either of constant complexity, in the case of immediate actions, either in $O(log(k))$, where k is the size of the queue, in the case of non immediate actions which necessitate to add the action in the priority queue. Of course, such a number depends on the configuration of the interactive score. At one extremity, we can consider scores *à la* Jean Haury, where each event is activated by the musician. The graphs that we obtain are very linear and in consequence, the number of actions in each place of the petri net is bounded by a small constant. Then, at the other extremity, let us consider a 4 hours piece whose source is the only interactive event. In this case, the place of the petri net that is associated to the source contains a very great number of non immediate actions, in fact, the activation of all events of the piece at the same transition. This is of course an extreme exemple whith a very low interest, but it shows that the system has to be improved in order to be efficient even in those kinds of configurations.

4. 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.).

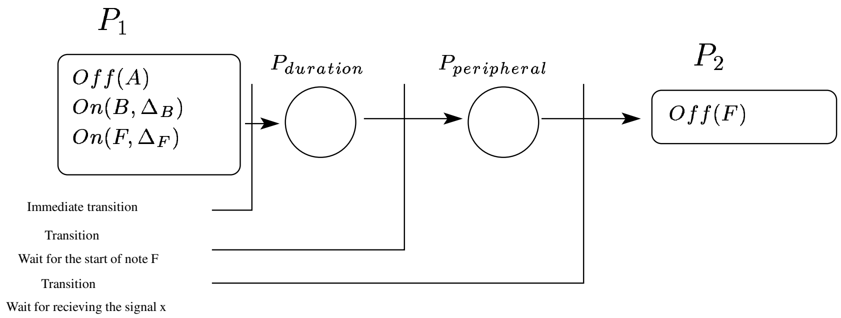


Figure 7. An example of a part of a Petri Network

Next step of this research consists in integrating 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.

5. REFERENCES

[All83] J.F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[BDC01] A. Beurivé and M. Desainte-Catherine. Representing musical hierarchies with constraints. In *Proceedings of CP'01, Musical Constraints Workshop*, 2001.

[DCB03] M. Desainte-Catherine and N. Brousse. Towards a specification of musical interactive pieces. In *Proc. of the CIM XIX, Firenze, Italy*, May 2003.

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

[Hau93] G. Haus. *Music Processing*. Oxford University Press, 1993.

[HS98] J. Haury and J. Schmutz. L'orchestre contre silence. In *Proc. of the JIM'98, Marseille, France*, pages D5–1, 1998.

[Mar02] A. Marsden. *Representing Musical Time - A Temporal-Logic Approach*. Swets and Zeitlinger, 2002.

[Mur89] T. Murata. Petri nets: Properties, analysis and applications. In *Proceedings of the IEEE*, volume 77(4), pages 541–580, April 1989.

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