A system of Interactive Scores based on qualitative and quantitative temporal constraints

Antoine Allombert, Myriam Desainte-Catherine, Gérard Assayag

A SYSTEM OF INTERACTIVE SCORES BASED ON QUALITATIVE AND QUANTITATIVE TEMPORAL CONSTRAINTS

A. Allombert
LaBRI/IRCAM

M. Desainte-Catherine, Joseph Laralde
LaBRI/Scrime

G. Assayag
IRCAM

ABSTRACT

We propose a formalism to compose and play interactive scores involving temporal structures and discrete interactive events. We use temporal logical constraints called the Allen's relations and also quantitative constraints on time intervals over temporal objects to define the global organization of the scores.

During the composition we maintain the constraints with a propagation model. During the performance we accept input controls to drive some temporal object and we have to maintain the constraints in a real-time context. At last, we present an example of a musical piece composed with a partial implementation of our model.

1. INTRODUCTION

To compose an interactive musical piece often necessitates to bind musical parts with interactive events or computing programs. But the musical writing systems propose limited real-time interaction while programming systems as Max don't provide sophisticated composition tools.

We claim that a new kind of systems providing composition environment as well as programming tools for specifying interaction is needed.

We focus on the question of interpretation of electroacoustic music which greatly deals with interaction. A composer writes a score by defining temporal elements with musical contents and then can define some specific controls that will be used by the perfomers during the excution to express through the piece. Tools allowing this type of composition/interpretion process for contemporary music are needed to explore new possibilities as shown in [7].

In this paper we propose a formalism for composing and performing musical pieces involving static and interactive events bounded by logical constraints. Here, we limit our study to temporal and duration constraints. We shall call this types of pieces *interactive scores* and the system based on this model *Iscore*. The first tests show this model to be appropriated to score editing and real-time requirements.

2. INTERACTIVE SCORES MODEL

Our model of scores has been presented in [3]. Thus, we will just recall here important notations and add some new definitions. A score is defined as a tuple $s =< t, r, l >$ where t is a set of temporal objects, r is a set of temporal relations and l is a set of linear constraints.

A temporal object is defined by $t =< s, d, p, c, n >$ where s is the start time, d is the duration, p is an attached process, c is a constraint attached to t (i.e. its local store) and $n = (t'_1 \dots t'_m)$ is a list of temporal objects embedded in t which are called the children of t. If n is empty, t is said to be a *simple* object otherwise it is a *complex* one.

We defined 5 classes of temporal objects :

- An interaction point has the constraint $d = 0$. Interaction points model discrete interactive actions. Its process consists in "listening" to the environment and waiting a triggering signal to happen.
- A texture has a generative process. It has the constraint $d \in [d_{min}, d_{max}]$ with $0 \leq d_{min} \leq d_{max}$, which gives an authorized range of variation. If $d_{min} = 0$ and $d_{max} = \infty$ then the texture is said to be *supple*, if we force $d = d_{min} = d_{max}$ then it is *rigid*, otherwise ($d_{max} \neq \infty$ and $d_{min} \neq d_{max}$) it is said to be *semi-rigid*.
- An interval is exactly like a texture except that it has no generative process. Intervals are blank $placeholders$ in the score ; they help to refine Allen relations
- A constraint object CO is a special type of interval which process consists in adding a set of global constraints into a global constraints store
- A control-point p is always created in relation with a texture/interval. Control points help to express any TO (Temporal Object) and a particular point inside a texture, an interval or a constraint object. As an interaction point, it has the constraint $d = 0$.

A temporal relation is defined by $r =< a, t_1, t_2 >$ where a belongs to A, the set of Allen relations [1] presented on figure 2, and t_1 and t_2 are temporal objects.

A linear constraint is defined by $l =< k, t_1, t_2 >$ where k is in $\mathbb{Q}$ and t_1 and t_2 are temporal objects. This forces the equation $d(t_1) = k.d(t_2)$.

Temporal objects : The composer can explicitly define Allen relations between TOs to bind them. Note that a *during* relation is automatically added between a TO and its children. The Allen relations are only qualitative while all initial temporal positions and durations are quantitatively specified in the score. But, the composer can also

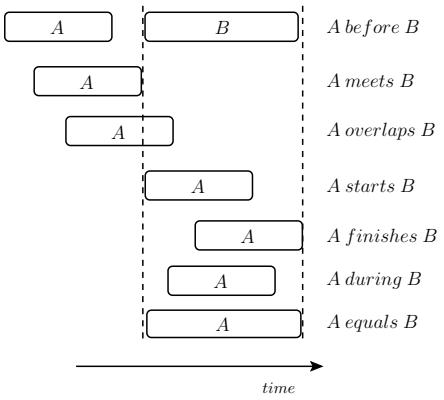


Figure 1. The Allen relations

add some quantitative constraints in specifying a TO to be *rigid* or *semi-rigid* or in adding linear constraints between TOs. We can see that the quantitative constraints will put restrictions on the Allen relations. Temporal relations are used to keep a global organization of the score whenever the composer changes the characteristics of TOs (start time, duration) at score edition time. The new values are propagated through the score and the TOs are moved or stretched to respect the constraints.

Interaction Points : We call an interaction point a particular event that is not to be played by the score player. It models a discrete, asynchronous event that is supposed to happen at performance time in the external environment and to input the system through an input channel. The composer can bind interaction points with any TOs through the *meets* relation. Therefore, the composer can express the way external controls will be able to drive the execution of the score at the performance time. Interaction points may happen at a certain distance from the date they are assigned to in the score, because of expressive choices or even mistakes. Thus, the point date in the score is the ideal date in the point of view of the composer and the Allen relations will be used to maintain the score coherence whatever the anticipation or the delay is. The composer can limit the range of anticipation and delay by using quantitative constraints. So the general philosophy behind this at performance time is "keep as much as possible the coherence of the time structure planned in the score, while taking into account, and accepting up to a certain limit the expressive freedom of the external agent".

Global Constraints : We allow the composer to define global constraints in order to catch specific constraints that cannot be expressed with unary or dual constraints as quantitative constraints or Allen relations. When a CO is added into the score, the global constraints he holds must be respected between its start time and its end time. Since COs are just specific TOs, the composer can synchronize the period of application of global constraints with any TO by using temporal relations. Its important to note that the objective of the global constraints is not to catch some special temporal relations because we want every temporal relations be expressed through Allen relations, durations constraints and linear constraints. The purpose of the global constraints is to catch constraints on the parameters of the process attached to the TOs which are out of the scope of this article.

At last, the graphical level provides a set of surface representations and graphical edition tools that may include conventional music notation (where it may apply) or hierarchical boxing representation such as in *OpenMusic* [4] and *Boxes* [5].

The figure 2 presents an interactive score.

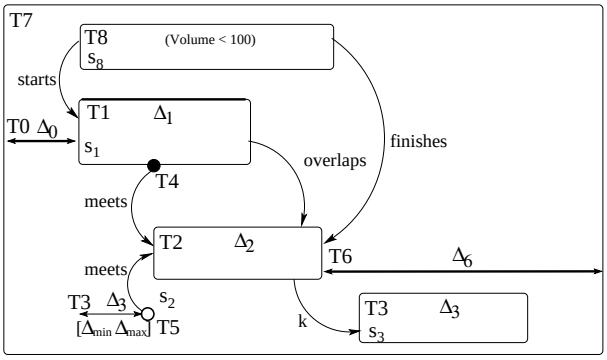


Figure 2. An example of interactive score

On this figure, we can find 8 $temporal\ objects$ from different types :

- several $textures$ (T_1, T_2, T_3 and T_7). Every $textures$ are simple ones except T_7. There are bounded to each others by $Allen\ relations$ ($overlaps$) and also by qualitative constraint ($d(T_2) = k.\Delta_3$). At last, the bold upside of T_1 means that T_1 is rigid so there is the constraint $d(T_1) = \Delta_1$.
- an $interaction\ point$ T_5 used to allow the interactive trigger of the start of T_2 by the performer during the execution.
- an $control - point$ T_4 used to synchronize the start of T_2 with a specific point or moment of T_1. This specific point is supposed to make sense with the process attached to T_1 (for example the climax of a dynamic variation).
- some $intervals$ (T_0, T_3 and T_6) that represent the specific status of some time intervals. T_0 and T_6 are rigid, then they force the two constraints :

$$s(T_1) - s(T_7) = \Delta_0$$

$$(s(T_7) + d(T_7)) - (s(T_2) + d(T_2)) = \Delta_6$$

  at last T_3 is semi-rigid and forces the constraint :

$$\Delta_{min} \leq s(T_5) - s(T_1) \leq \Delta_{max}$$

- a $constraint\ object$ T_8 bounded to the $textures$ T_1 and T_2 with $Allen\ relations$. It forces a global

constraint on the volume. As it is a global constraint, this volume is the volume of the mix of the textures that sets during T_8, i.e. T_1 and T_2 and eventually T_3 if the performer triggers T_5 (the start of T_2) enough late so T_2 overlaps T_3.

We have implemented into *OpenMusic* a version of the interactive scores model limited to the case where every TOs are *supple* and where there is no linear constraints. The composer can only define Allen relations and global constraints. The figure 3 presents some screenshots of our implementation of *Iscore*.

There are two different steps in the interactive scores model : the edition time and the performance time. During the edition we have to support the add and remove of constraints and maintain these ones when the composer changes some characteristics while during the performance we have to take into account the performer choices and to maintain the constraints under a strong real-time constraint.

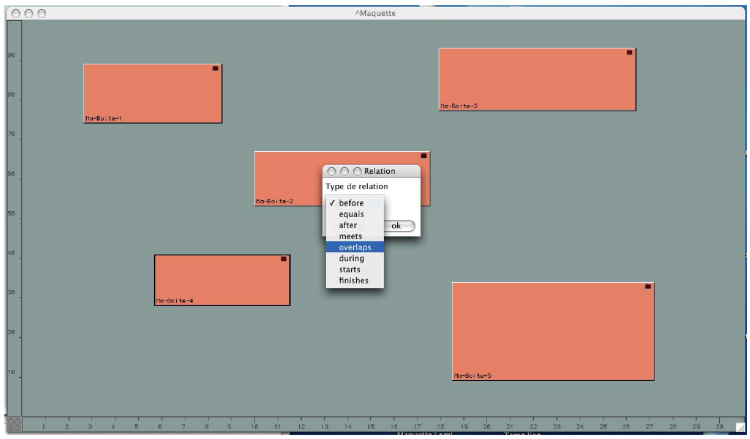
Since computation time is not critic during the edition time, we use a general constraints solver called Gecode [10] which propagates the new relations, constraints and values. This cannot be done in a real time context since we are not able to control the computation time.

3. REAL-TIME MODEL

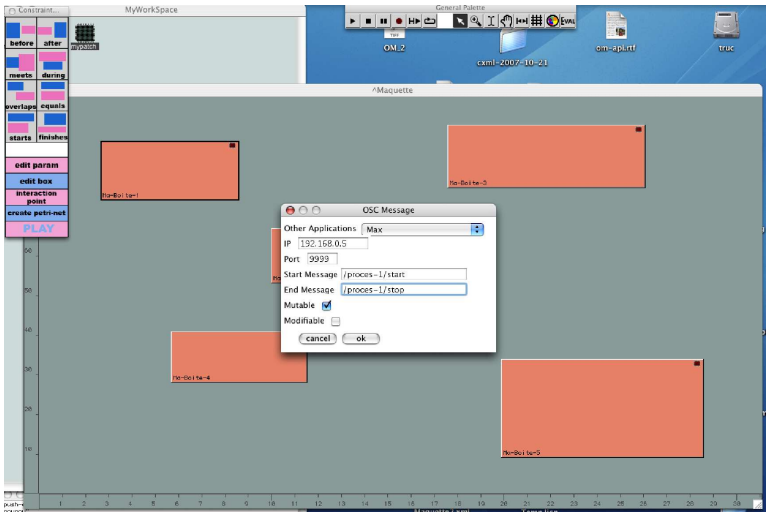
We proposed a real-time model in [2] for a limited model of interactive scores. In this case, the temporal constraints in the scores are limited to the Allen relations. So there are no $intervals$ and all the $textures$ are $supple$. This model is based on Petri nets to which we add a global constraints store. A Petri net is a specific states machine which can run concurrent processes that must synchronize at particular moments. Formally, it is a bipartite directed graph with two types of vertice called “places” and “transitions”. An edge can only connect two vertices from different types. At last places contain a number of tokens greater or equal to zero. A state of the system is represented by a distribution of tokens among the places. The system changes his state by crossing “transitions”, this modifies the tokens distribution by a consummation/production process. In our case, we use a special type of Petri nets called *Time Petri Nets* [9] which allows to associate a time-range of wait in places before crossing the transitions.

After the edition time, we transform the interactive score into a Petri net by associating *events* with *transitions* while *places* are used to wait between *events*. We represent the Allen relations through edges connections. For example if there is the relation $r =< before, t_1, t_2 >$, E_{t_1} represents the transition associated to the end of t_1 and S_{t_2} the one associated to the start of t_2, we will find in the net a place P_r representing the time to wait between the end of t_1 and the start of t_2 with an edge from E_{t_1} to P_r and an other one from P_r to S_{t_2}. So we will be sure that the relation r is maintained.

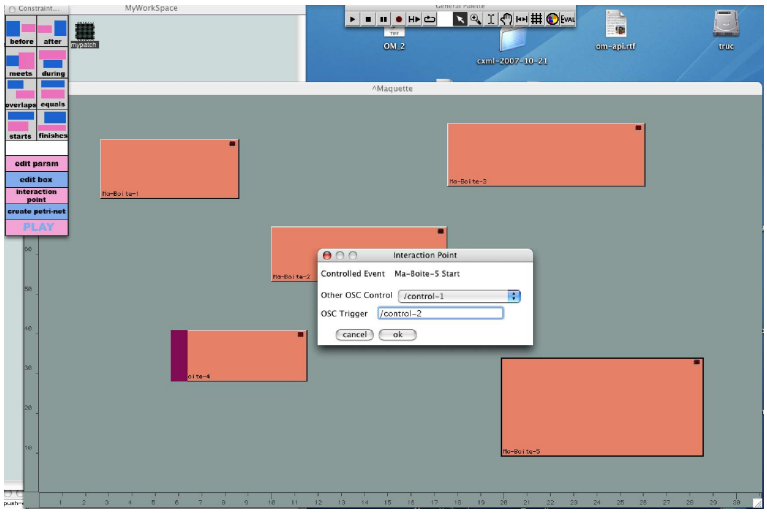
In addition, we don’t use fixed delays in places but ranges to permit synchronizations. When a token is cre-



(a) Adding a Allen’s relation



(b) Defining the OSC messages associated with the boxes



(c) Adding an interaction point

Figure 3. Some screenshots of *Iscore* in Open Music

ated in a place P with a range $[t_{min}, t_{max}]$, an intern timer t is launched from 0. While $t \leq t_{min}$, the token cannot be consummated (this means that the transition that follows P cannot be crossed). When $t = t_{max}$, the system crosses the following transition if it can. Formally, the range forces the following transition of P to be crossed when $t_{min} \leq t \leq t_{max}$. If this transition waits for an external event to come as for a transition representing an interactive event, then the system doesn't take into account the performer's actions before the timer t reaches the value t_{min} and if the performer doesn't trigger the crossing of the transition before t reaches t_{max}, then the system automatically crosses the transition.

It is important to note that for a transition T, it can exist several places P_i with an arc $A_i = Pi \rightarrow T$. Each T_i has a range $[t_{min_i}, t_{max_i}]$ and a timer t_i. In this case, T can be crossed when :

$$\forall i, t_{min_i} \leq t_i \leq t_{max_i}$$

This type of synchronization situation can lead to critical situation if :

$$\exists i, j | t_i < t_{min_i} \; and \; t_{max_j} < t_j$$

This means that T should be crossed to respect one of the ranges while it cannot be crossed according to an other range. In our system, we always try to prevent this kind of situation.

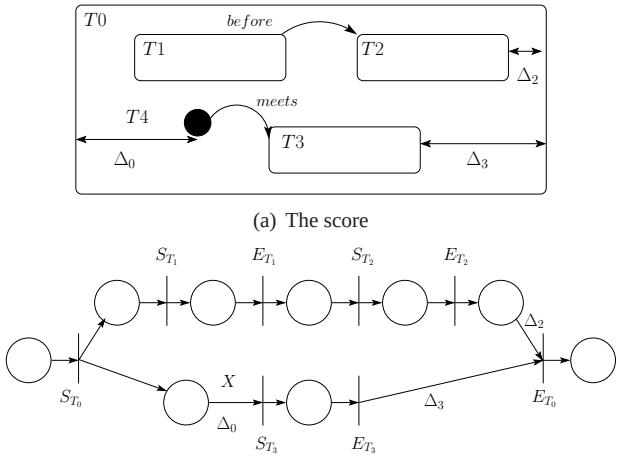
In the limited system, we use 2 different temporal ranges in the places : if the place represents the wait of an external control, its range is $[0, \infty]$ since we can accept any anticipation or delay (every TOs are *supple*) ; for the other places, the range is $[\Delta_{score}, \infty]$ where Δ_{score} is the value written in the score for the wait duration represented by the place. The value ∞ translates the fact that a wait duration can be increased by an interactive event not directly connected to it but which can influence the wait duration through synchronization configurations. Thus if no *interactive events* influences the wait duration, its value is Δ_{score} else it is a greater one.

We can find an example of the transformation of a very simple score into a Petri net on the figure 4.

In this example, there are 4 $textures$ and an *interaction point* T_4. On the Petri net, the symbols s_{T_i} and e_{T_i} denotes the events *start of* T_i and *end of* T_i. The Allen relations $T1 \; during \; T0$, $T2 \; during \; T0$, $T3 \; during \; T0$ and $T_1 \; before \; T2$ are represented by the configuration of the Petri net. The symbol X denotes the external control used to trigger T_4. At last, we lay emphasis on 3 intervals :

- Δ_0 between $s(T_0)$ and $s(T_3)$
- Δ_2 between $e(T_2)$ and $e(T_0)$
- Δ_3 between $e(T_3)$ and $e(T_0)$

In the Petri net the range of Δ_0 is $[0, \infty]$ since Δ_0 is $supple$ and we allow the performer to trigger the control X whenever he wants. The transition representing $e(T_0)$



(a) The score

(b) The associated Petri net

Figure 4. An example of the transformation of a score

is typically a synchronization transition. In this case, the ranges of Δ_2 and Δ_3 are $[\Delta_{2_{score}}, \infty]$ and $[\Delta_{3_{score}}, \infty]$. This permits the synchronization. In fact, the modification of the date of $s(T_3)$ by the performer will propagate to $e(T_3)$. Since the date of $e(T_3)$ may be modified, then values of Δ_2 and Δ_3 may also be modified. With the ranges we defined, we ensure that :

$$\Delta_{2_{score}} \leq \Delta_2$$

$$\Delta_{3_{score}} \leq \Delta_3$$

More precisely, if $s(T_3)$ is anticipated, Δ_3 will increased while $\Delta_{2_{score}}$ will be respected. On the contrary, if $s(T_3)$ is delayed, Δ_2 will increase and $\Delta_{3_{score}}$ will be respected.

At last, the composer can change the default ranges of intervals preceding a synchronization transition. He can choose $[0, \infty]$ for some intervals. This implies that the values $\Delta_{i_{score}}$ whith such a range will not be ensured. Then the composer can give a priority on intervals that he wants to last at least the written values while the other can be totally modified.

4. QUANTITATIVE CONSTRAINTS

Now we want to take into account the quantitative constraints. For the edition time we still use a constraints solver such as Gecode, since we don't care about the computation time. But we have to modify our real-time model.

In fact, to introduce qualitative constraints rises up the possibility for the performer to break some constraints of the score during the execution. Our aim is to ensure that the score will bee totally played without any broken constraints. This aim is quite difficult to reach since we have to anticipate the actions of the performer that could lead to inconsistent situations.

The figure 5 shows a an example of this type of situation. We can see a *rigid* complex texture P ($d(P) = \Delta_P$) with two children A and B, a relation $r =< before, A, B >$

and an interactive event T_i controlling the beginning of A. So, the value Δ_1 can change under the influence of T_i. Since we have the equation : $d(P) = \Delta_1 + d(A) + \Delta_2 + d(B) + \Delta_3$ it is clear that values $d(A)$, $\Delta_2, d(B)$ and Δ_3 may be modified to maintain $d(P) = \Delta_P$. In the example, T_i is delayed so we have to reduce the durations values. There are several ways to compute these values, so we guided the computation in adding an order of reduction chosen by the composer at composition time among 2 alternatives : left reduction which means that the reduction order is the chronological order (the example choice) or a right reduction which is the reverse order of the first. Since duration values may change, we have to modify the time-ranges associated to the places of the Petri net.

Encoding the quantitative constraints directly in the Petri nets appeared to be inapplicable in the general case. As a consequence, we have to add to the petri net that holds the Allen relations, a constraints system or CSP (consraints satisfaction problem) that holds the quantitative constraints. This CSP is from the same type as the one we use during the edition step to maintain all the constraints. The difference with the edition step is that during the execution, we have the petri net that maintains the Allen relations without constraints computation. Then, we want to put into this CSP only the constraints that cannot be hold by the Petri net. One can also see that some specific quantitative constraints can be very easyly represented in the Petri net. For example, an interval between a satic event and an interactive one that follows it (such as Δ_0 on the figure 4), can be forced to be $semi - rigid$ with the range of values $[Val_{min}, Val_{max}]$ simply by using this range in the Pretri net in the place that represents this interval. It is very important to deal with the smallest CSP as possible to prevent from excessive computation times, then we have to clearly identify the constraints that cannot be represented in the Petri net.

To represent this CSP, we use a constraints graph which is a bi-partite graph in which variables and constraints are represented by the vertice and there is an arc between a vertex labeled by variable a V and the vertex labeled by a constraint C only if V is involved in C in the CSP. We want to use this representation with a propagation algorithm that allow us to propagate the dates of the interactive events over the CSP and compute new values for the textures and intervals durations that take into account the actions and choices of the performer.

To determine which intervals will appear into the graph, we define for each interactive event e a durations set Z_e called the $influence\ area$ of e including all the durations that could be modified by e through qualitative constraints. Eventually the variables set of the constraints graph associated to a score s will be $\bigcup_{e \in s} Z_e$ where the durations are seen as the variables representing them. This means that the constraints graph will only include the intervals that may change during the execution.

Each variable of the graph is associated to a *domain* which is a range of values $[Val_{min}, Val_{max}]$. The domain D of a variable var is the set of values such as for

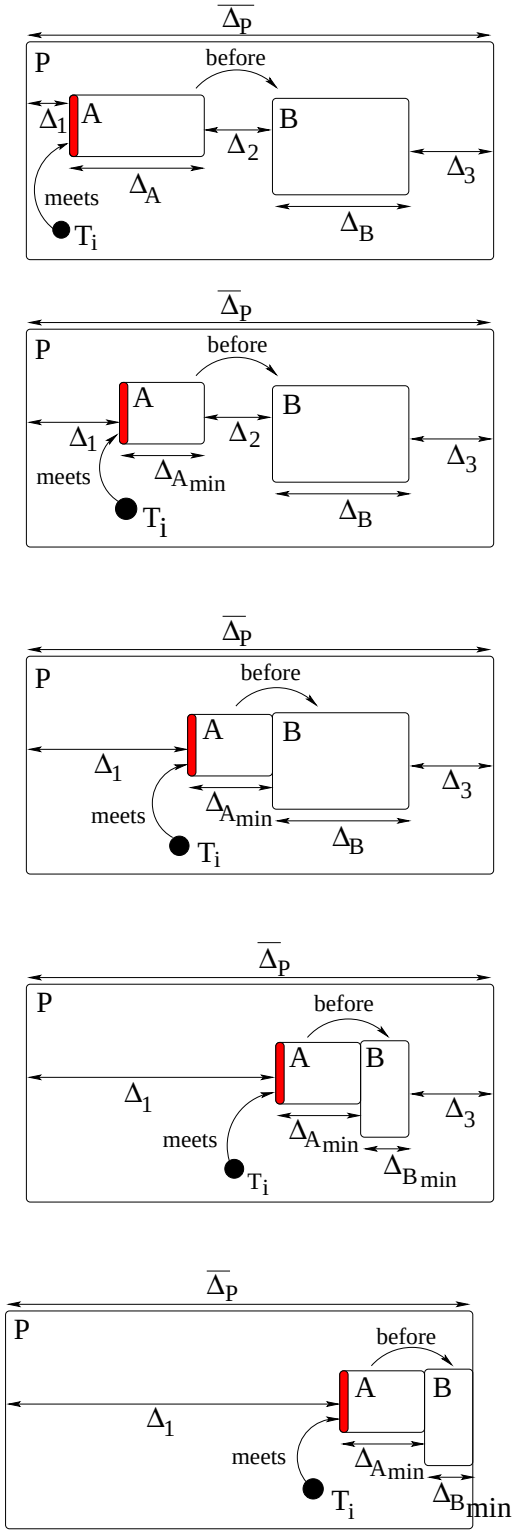


Figure 5. An example of reduction with left priority

val in D, there is a solution to the constraints problem accepting $var = val$. Computing the variables domains of a general constraint problem is not easy. In our case we use a domain reduction algorithm inspired by *Indigo* [6] which reduces the domains from the initial range $[0, \infty]$.

After the edition time, we turn the score into a Petri net and a constraints graph. Before the performance we run the domains reduction algorithm to compute the time-ranges of the arrival of control inputs, because we want them such as whatever the anticipations or delays on interactive events there will be no incoherence with the quantitative constraints. During the performance we run the reduction algorithm each time an interactive event happens to reduce the variables according to the anticipation or delay of this event. Of course, the domain of an interval in the constraints graph will be same as the time range we use in the Petri net for this interval. Then, each time we recompute the domain of an interval with the propagation algorithm, we set the corresponding time range in the Petri net to the new domain.

Unfortunately, the reduction algorithm is efficient only over acyclic constraints graphs which is not the general case. If the graph is cyclic the domains after the reduction algorithm can contain some values that lead to an incoherence. This is particularly crucial for the domains of intervals involving interactive events because this means that when the performer will have to trigger such an event, we may accept some values that lead to inconsistent situations. One solution could be to test each value of the domains after the use of the reduction algorithm but this one is clearly too expensive in time.

To prevent from this expensive computation we are thinking about only test the extreme limits of the domains after running the propagation algorithm and use a property of convexity of the domains. That is for a variable V with a domain D and v_1 and v_2 in D :

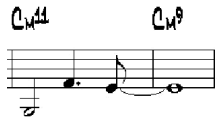
$$\forall v | v_1 \leq v \leq v_2, v \in D$$

Thus our idea is, after running the propagation algorithm on a cyclic graph, to test the values Val_{min} and Val_{max} for the domains of the form $[Val_{min}, Val_{max}]$. If solutions exist with this values, we accept the domain, if not, we recompute the domains. This solution is still on the drawing board.

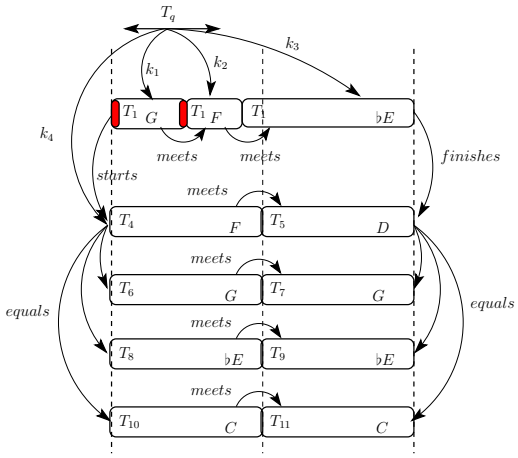
But even if the acyclic graphs are not the general case, they present interesting cases.

5. AN EXAMPLE WITH ACYCLIC GRAPH

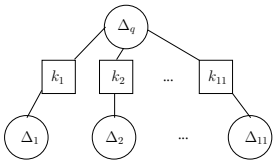
Acyclic constraints graphs can be found in some interesting cases. The example presented on figure 6 consists in interactively setting the tempo of 2 bars of the bossa-nova standard Blue Bossa by Kenny Dorham. We turn the 2 bars into an interactive score with temporal objects representing the notes, Allen's relations designing the temporal organization and an interval T_q representing a time-unit which is related to each duration by a linear constraint. These constraints express the durations in time-unit. T_1 has interactive start and end, so during the performance $d(T_1)$ will be changed. Using the constraints graph, we will propagate this modification to $d(T_q)$ and then to all the other durations in order to simulate a change of the tempo. With similar configurations, one can define automatic "accelerendo" or "descelerendo".



(b) 2 bars of Blue Bossa



(c) The 2 bars written as interactive score



(d) The associated constraints graph

Figure 6. An example of an interactive tempo setting

6. ECO MACHINE

We introduced different structures to represent the scores during the execution : petri net, constraints graph, store of global constraints... This structures are bounded together and must be jointly used during the performance step. Then we gather all this structures in what we call a *musical environement* that will be executed during the performance. Formally, this environement will interpreted by an abstract machine that we call the and that we introduced in [8]. This is an abstract machine such that :

- a state of the ECO machine is a tuple (E, C, O, t) where :
  - E is a musical environement
  - C is a control string representing time-stamped events
  - O is the output string

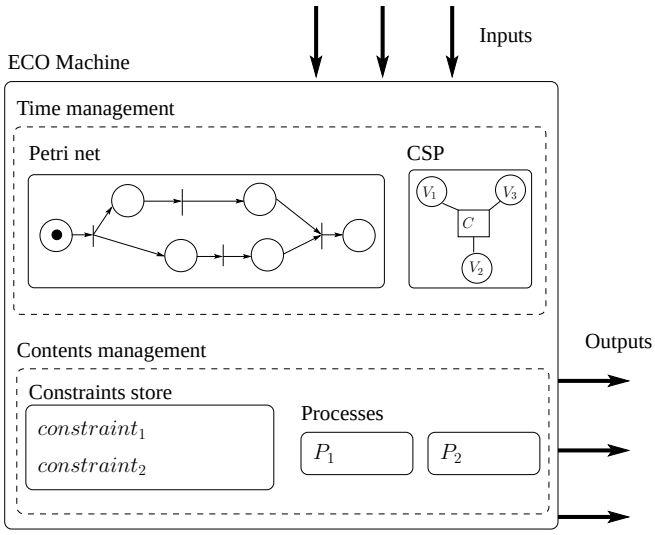


Figure 7. The ECO Machine

– t is the time-stamp of the state

- the operation of the machine is described in terms of state transitions that synchronized on a clock.

The figure 6 presents the schema of the ECO machine in a particular state during the execution. As we can see, the musical environement contains all informations needed to produce the outputs. We can see that we separate the *musical envornement* in two parts, one deals with the temporal structures and the other deals with the processes attached with the temporal objects and their parameters. Then after the edition step, we turn the score into a *musical environement* that will represent the temporal and musical informations and will be interpretable by the *ECO machine*. This abstract machine is generic and will be stricly the same for each score, while the *musical environement* will depend on the score.

During a step of the execution of the machine, the following operations occur :

- watch the inputs from the control string C
- cross transitions of the Petri net under the action of incoming inputs or depending on the time intervals. Trigger the events associated to this transitions.
- ask the global constraints store if the parameters of the processes don't break a global constraint, if so, re-compute some parameters that respect the global constraints.
- starting or stopping processes with right parameters and send the result on the ouputs.

7. A COMPOSITION EXAMPLE

Here we present an example of a musical piece called Le Pays du Soleil Couchant written with the partial implementation of *Iscore* into *OpenMusic* by Joseph Laralde, musical assistant at Scrime [1]. In this implementation, OSC messages can be defined to be sent when temporal objects start and end. The interactive events are also triggered by OSC messages.

The figure 8 presents a part of the score. We can see that there is an instrumental part which controls some interactive events through control pedals and that the temporal objects are used to communicate with Max/MSP. Precisely, the first object is used to control the recording of samples that are used in some granular synthesis processes controlled by the other temporal objects.

According to Joseph Laralde : "The main interest in i-score is that it perfectly completes Max/MSP as a sequencer with a real-time approach. I can receive interaction messages from Max/MSP that can be information from MIDI devices as well as real-time sound analysis results and send it back osc messages to trigger the events I need. This could be done only with Max/MSP but with much more difficulty and the result would be quite messy and hard to develop at the same time the piece is being written."

8. CONCLUSION AND FUTURE WORK

We presented in this paper a system for composing interactive scores and playing them. We used Allen relations and durations constraints to define temporal organization and specific models using constraints propagation and Petri nets to maintain these constraints. We presented examples of the use of interactive scores including an original composition. The next step of this work will be to find a solution for scores with cyclic constraints graph and to continue implementing and testing *Iscore*. We are also involved into a project for adapting our model to the needs of theater stage managers.

9. REFERENCES

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

[2] A. Allombert, G. Assayag, and M. Dessainte-Catherine. A model of interactive scores based on petri nets. In *Sound and Music Computing, SMC 2007*, 2007.

[3] A. Allombert, G. Assayag, M. Dessainte-Catherine, and C. Rueda. Concurrent constraints models for interactive scores. In *Sound and Music Computing, SMC 2006*, 2006.

[4] G. Assayag and al. Computer assisted composition at ircam : From patchwork to openmusic. *Computer Music Journal*, 23(3), 1999.

[1] Studio de Création et de Recherche en Informatique et Musique Electroacoustique, Bordeaux

[5] A. Beurivé. Un logiciel de composition musicale combinant un modèle spectral, des structures hiérarchiques et des contraintes. In *Journées d'Informatique Musicale, JIM 2000*, 2000.

[6] Alan Borning, Richard Anderson, and Bjorn N. Freeman-Benson. Indigo: A local propagation algorithm for inequality constraints. In *ACM Symposium on User Interface Software and Technology*, pages 129–136, 1996.

[7] K. Dahan and M. Laliberté. Réflexions autour de la notion d'interprétation de la musique électroacoustique. In *Pr. of* 13^{ieme} *Journées d'Informatique Musicale (JIM 08), Albi, France*, March 2008.

[8] M. Desainte-Catherine and A. Allombert. Interactive scores : A model for specifying temporal relations between interactive and static events. *Journal of New Music Research*, 34(4), December 2005.

[9] P.M. Merlin. A study of the recoverability of computing systems, 1974.

[10] C. Schulte and G. Tack. Views and iterators for generic constraint implementations. In *International Colloqium on Implementation of Constraint and Logic Programming Systems, CICLOPS05*, 2005.

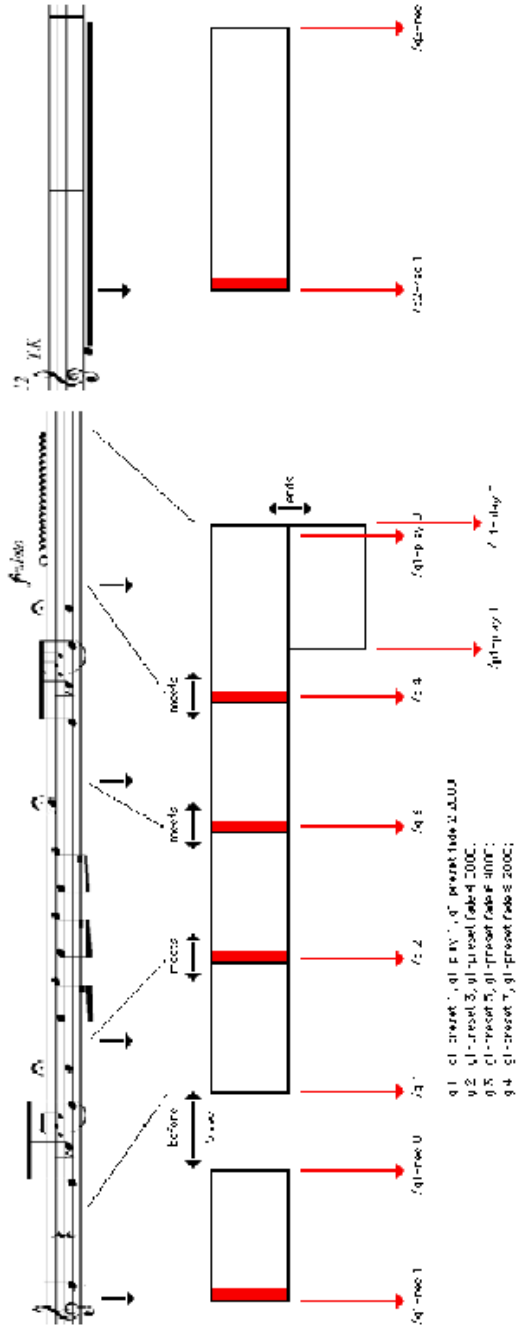


Figure 8. An extract of the score of *Le Pays du Soleil Couchant*