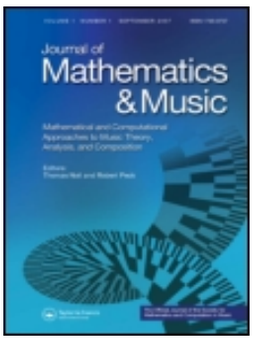

Taylor & Francis
Taylor & Francis Group

Journal of Mathematics and Music

Mathematical and Computational Approaches to Music Theory, Analysis, Composition and Performance

ISSN: 1745-9737 (Print) 1745-9745 (Online) Journal homepage: https://www.tandfonline.com/loi/tmam20

Formal semantics for interactive music scores: a framework to design, specify properties and execute interactive scenarios

Mauricio Toro, Myriam Desainte-Catherine & Camilo Rueda

To cite this article: Mauricio Toro, Myriam Desainte-Catherine & Camilo Rueda (2014) Formal semantics for interactive music scores: a framework to design, specify properties and execute interactive scenarios, Journal of Mathematics and Music, 8:1, 93-112, DOI: 10.1080/17459737.2013.870610

To link to this article: https://doi.org/10.1080/17459737.2013.870610

Journal of Mathematics and Music, 2014
Vol. 8, No. 1, 93–112, http://dx.doi.org/10.1080/17459737.2013.870610

Taylor & Francis
Taylor & Francis Group

Formal semantics for interactive music scores: a framework to design, specify properties and execute interactive scenarios

Mauricio Toro[a*], Myriam Desainte-Catherine[a] and Camilo Rueda[b]

[a]*LaBRI, UMR 5600, SCRIME, IPB, ENSEIRB-Matmeca, Université de Bordeaux, Talence, France;*
[b]*Pontificia Universidad Javeriana Cali, Cali, Colombia*

(*Received 15 July 2011; accepted 26 November 2013*)

Most interactive scenarios are based on informal specifications, so that it is not possible to formally verify properties of such systems. We advocate the need for a general and formal model aiming at ensuring safe executions of interactive multimedia scenarios. Interactive scores (IS) is a formalism based on temporal constraints to describe interactive scenarios. We propose new semantics for IS based on timed event structures (TES). With such a semantics, we can specify more properties of the system, in particular, properties about execution traces, which are difficult to specify as constraints. We also present an operational semantics of IS based on the non-deterministic timed concurrent constraint calculus and we relate such a semantics to the TES semantics. With the operational semantics, we can describe the behaviour of scores whose timed object durations can be arbitrary integer intervals.

Keywords: event structures; ccp; temporal constraint programming; interaction; ntcc; concurrent constraint programming; interactive music scores; concurrency; synchronization

CR Category Numbers: D.1.6; E.4; H.5.5; J.5

1. Introduction

Interactive multimedia deals with the design of scenarios where multimedia content and interactive events can be handled by computer programs. Examples of such scenarios are multimedia installations, interactive museum exhibitions and some Electroacoustic music pieces. Designers usually create multimedia content for their scenarios, and then bind them to external interactive events controlled by *Max/*MSP programs (Puckette, Apel, and Zicarelli 1998). We advocate a model that encompasses facilities (i) to design multimedia scenarios having complex temporal relationships among components and (ii) to define effective mechanisms for synthesis control based on human gestures.

Such a general model must have formal semantics, as required for automated verification of properties of the scenario that are fundamental to its designers and users. As an example, to verify that timed objects (TO) will be played as expected during performance. In general, we want to prove some property of each execution trace; for instance, we want to prove that a music motive with notes C-D-E appears in all the traces of execution (or at least in one). Such properties cannot

*Corresponding author. Email: mtoro@cs.ucy.ac.cy

be verified in applications based on informal specifications, as it is the case for most existing scenarios with interactive controls. Limited reusability is also a problem caused by the lack of formal semantics: A module made for one scenario might not work for another one because the program may have dependencies on external parameters that are not stated explicitly.

Fortunately, there are formalisms to model interactive scenarios such as *interactive scores* (IS). IS has been a subject of study since the beginning of the century (Desainte-Catherine and Brousse 2003). The first tool developed was *Boxes* (Allombert, Desainte-Catherine, and Assayag 2008). Boxes was conceived for the composition of Electroacoustic music with temporal relations; however, user interaction was not provided. A recent model of IS (Allombert 2009), that significantly improves user interaction, has inspired two applications: *Iscore* (Allombert, Assayag, and Desainte-Catherine 2008) to compose and perform Electroacoustic music and *Virage* (Allombert et al. 2010) to control live performances and interactive exhibitions.

Scenarios in IS are represented by *TOs*, *temporal relations* and *interactive objects*. Examples of TOs are sounds, videos and light controls. TOs can be triggered by interactive objects (usually launched by the user) and several TOs can be executed simultaneously. A TO may contain other temporal objects: this hierarchy allows us to control the start or end of a TO by controlling the start or end of its parent. Hierarchy is ever-present in all kinds of music: Music pieces are often hierarchized by movements, parts, motives, measures, among other segmentations. Moreover, Vickery argues that, in interactive music, a hierarchy is convenient to control higher-order parameters of the piece; for instance, to control the volume dynamics instead of the volume of each note (Vickery 2004).

Temporal relations provide a partial order for the execution of the TOs; for instance, temporal relations can be used to express precedence between two objects. In IS, it is also possible to specify a variety of relations among TOs such as harmonic relations, global conditions and conditional branching. In this paper, we take into account IS limited to hierarchical relations represented as a directed tree and *point-to-point temporal relations* without disjunction nor inequality ($\neq$) (Gennari 1998). Gennari (1998) proved that point-to-point relations are equivalent to *interval-to-interval Allen's relations* without disjunction (Allen 1983) (e.g. a during b, a overlaps b and a meets b). Allen's relations were previously used in IS by Allombert et al. (2006) and Allombert, Assayag, and Desainte-Catherine (2008).

In Allombert et al.'s models of IS, it was not precisely stated how to execute scores whose TO durations are arbitrary integers intervals; for instance, a score in which object a must be executed between two and four time units after object b. These works handle *flexible-time integer intervals*: $\{0\}$ to express simultaneity, and $\{1, 2 \ldots\}$ and $\{0, 1 \ldots\}$ for precedence. Allombert et al.'s models also miss an abstract semantics.

Furthermore, representing the duration of a TO as a set of integers, and not necessarily as an integer interval, allows us to model rhythmical patterns; for instance, that a music object should be played at beats one, three or five (but not two nor four). Constraints of this form are found in the improvization system presented by Rueda and Valencia (2004).

In Allombert et al.'s model (Allombert 2009), IS are defined using recursive structural definitions: A TO is a tuple that contains a set of temporal objects. Those definitions pose difficulties as a formal semantics of IS; for instance, awkward conditions would have to be added to those definitions just to specify the hierarchy as a directed tree. For that reason, we model a TO as a directed tree.

As mentioned before, in our model a TO comprises two points, called the *starting* and *ending* points, together with a set of integers called the *duration* set of the object. Each value in this set is a possible *choice* for the distance between the starting and ending points of the object. In the implementation of a TO this distance is interpreted as a number of *time units*: each of the values in the set represents thus one possible choice for the number of time units that must elapse between the starting and ending of some *activity* associated with the object. In a TO the activity

together with its starting-ending points and duration set is called a *box*. A TO can be thought as a boxes-within-boxes hierarchy.

In our model, starting-ending points of TOs are mapped to points on a time line. This mapping could be fully or partially determined. Temporal relations constrain the set of mapping possibilities. A partial order among TOs is given by *point-to-point* relations; for instance, the start of *a* is executed five time units before the end of *b*.

We provide an abstract semantics for IS based on *timed event structures* (TES) (Baier, Katoen, and Latella 1998). The purpose of such a semantics is (i) to provide an easy, declarative way, to understand the behaviour of a score, and (ii) a simple theoretical background to specify properties of the system. In constraint programming, we can specify some properties of the scores such as playability. We can also specify those properties in event structures; moreover, the notion of *trace*, inherent in event structures, is more appropriate than temporal constraints for certain properties. As an example, to specify that a music motive appears in at least one trace of execution.

This study led us to discover that there is no difference between interactive objects and the other TOs in the event structures semantics: such a difference can only be observed in the operational semantics. That was the main reason to introduce an operational semantics based on non-deterministic timed concurrent constraint (`ntcc`), along the lines of Rueda et al. concurrent constraint model in Allombert et al. (2006).

In this paper, we extend the IS formalism with an abstract semantics based on event structures and an operational semantics specified in the `ntcc` calculus (Nielsen, Palamidessi, and Valencia 2002), providing (i) a new insight into the IS model; (ii) more complex temporal relations to bind objects, including arbitrary sets of integers in the event structures semantics and arbitrary integer intervals in the operational semantics; and (iii) the possibility to verify properties over the execution traces. In order to use arbitrary integer intervals in our operational semantics, we show that several transformations to the event structures semantics are needed to define operational semantics that can *dispatch* the TOs of the score in real-time.

We also chose event structures because it is a powerful formalism for concurrency that will allow us to extend the IS semantics with conditional branching and loops in a very precise and declarative way. Conditional-branching timed IS were introduced in Toro and Desainte-Catherine (2010). Such an extension has operational semantics based on `ntcc`, but it misses an abstract semantics to understand the conflicts among the TOs that take place when modelling conditions and choices.

`Ntcc` is a process calculus that models concurrency, non-determinism and asynchrony. We show an equivalence between the event structures semantics and the operational semantics. We believe that we can run the `ntcc` model in *Ntccrt*, a real-time capable interpreter for `ntcc` (Toro et al. 2009).

A similar approach to our `ntcc` model is proposed by Olarte et al. to change the hierarchy of TOs during execution (Olarte and Rueda 2009). The spirit of this approach is different: it focuses on changing the structure of the score during execution to allow the user to 'improvise' on a written piece, whereas we are interested in a simpler model that we can execute in real-time. Another system dealing with a hierarchy of TOs is *Maquettes of OpenMusic* (Bresson, Agón, and Assayag 2005). Nonetheless, *OpenMusic* is a software for composition and it is neither meant for real-time interaction.

The remainder of this paper is structured as follows. In Section 2, we introduce IS. In Section 3, we introduce event structures. In Section 4, we define the event structures semantics of IS. In Section 5, we formalize some properties of IS. In Section 6, we introduce the intuitive semantics of `ntcc` and we define the operational semantics of IS. Finally, in Section 7, we give some concluding remarks and we contrast our semantics with the models previously published by Allombert et al.

2. Interactive scores

IS are composed of *temporal relations* and *TOs*. Intuitively, a TO is a collection of boxes arranged in a hierarchical structure. Each box represents some meaningful activity in a musical piece; for instance, playing a note, triggering a synthesis process or waiting for a musician interaction. The starting and ending of each box activity is identified by two *points*. The hierarchical structure of a timed object can be seen as a tree in which the nodes are the boxes of the TO and the arcs represent hierarchical relations: an arc between b_1 and b_2 means that the TO whose root box is b_1 'contains' the boxes in the subtree rooted by b_2.

DEFINITION 2.1 (TO) *Let P be a set of points. A TO $\langle I, H\rangle$ is a directed tree with nodes I and arcs $H \subseteq I \times I$. Elements of I, the boxes, are tuples $\langle sp, ep, \Delta, sa, ea\rangle$, where $sp, ep \in P$, are called start and end points, respectively, $\Delta \subseteq \mathbb{N}_0$ is a set of possible durations, and $sa, ea \in Act$ are the start and end activities, respectively. A TO whose root box has $\Delta = \{0\}$ and has no subtrees is called an interactive object. All points of the boxes in a TO are pairwise distinct. The set of all TOs is $\mathcal{T}$.*

In musical applications, TOs are defined together with a function $\nu : P \to \mathbb{N}_0$ mapping points in boxes into a time line. For a point $p \in P$, we call $\nu(p)$ the *time position* of p. We thus think of Δ above as a set of (non-negative integer) possible temporal *distances* between the starting and ending time positions of points of a box. The set Δ thus can be seen to define a condition on the temporal positions map. We are interested in providing ways to further constrain the map; for instance, to define relations for the distance between the starting points of two different boxes.

For any $t \in \mathbb{N}_0$ and $\Delta \subseteq \mathbb{N}_0$, we define $t \oplus \Delta \stackrel{\text{def}}{=} \{t' | t' = t + \delta, \delta \in \Delta\}$. Let $\nu : P \to \mathbb{N}_0$ be a time positions map. For any box $\langle p, q, \Delta, sa, ea\rangle$ in a TO, ν must satisfy $\nu(q) \in \nu(p) \oplus \Delta$ The collection of these constraints on maps for all boxes in a TO specify the set of *potential time positions* for each point in them.

DEFINITION 2.2 (Temporal relation) *A temporal relation is a tuple $\langle p, \Delta, q\rangle$, where $p, q \in P$ and $\Delta \subseteq \mathbb{N}_0$. The set of all temporal relations is $\mathcal{R}$.*

DEFINITION 2.3 (IS) *An IS is a TO equipped with a set of temporal relations: A tuple $\langle o, R\rangle$, where $o = \langle I, H\rangle$ is a TO and $R \subseteq \mathcal{R}$ is the set of temporal relations that includes at least*:

(1) *For each $i = \langle sp, ep, \Delta\rangle$ such that $i \in I$, $\langle sp, \Delta, ep\rangle \in R$*
(2) *For each $(i_1, i_2) \in H$, $\langle \text{sp}(i_1), \{0, 1 \ldots\}, \text{sp}(i_2)\rangle \in R$ and $\langle \text{ep}(i_2), \{0, 1 \ldots\}, \text{ep}(i_1)\rangle \in R$*

An IS further constrains time positions of points in its temporal object. Let $IS = \langle o, R\rangle$ be an IS. A time positions map $\nu : P \to \mathbb{N}_0$ for IS is *viable* if, for every $\langle p, \Delta, q\rangle \in R$, condition $\nu(q) \in \nu(p) \oplus \Delta$ holds. For simplicity and when no confusion arises, we replace $\oplus$ by $+$ and write $n + \Delta$, for $n \in \mathbb{N}_0, \Delta \subseteq \mathbb{N}_0$

Figure 1 is an example of a score; its structural definition and temporal constraints are presented in Appendix 1. As a music example, consider that the object *sound* presented in Figure 1 is the one described in Figure 2.

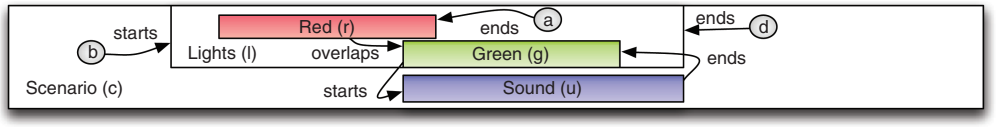

Figure 1. Example of a score with hierarchy. For simplicity, we use Allen's *overlaps* relation. This relation can be easily encoded into point-to-point relations, as described in Figure 5.

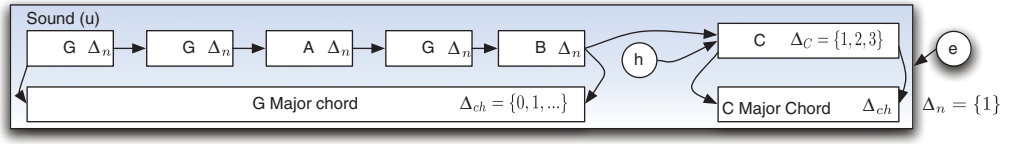

Figure 2. Example of the *sound* TO in Figure 1. Arrows represent temporal relations labelled by $\Delta = \{0\}$. There is a sequence of notes G,G,A,G,B,C accompanied by the chords G *Major* and C *Major*. The duration of the last note (C) is determined by an interactive object *h* that controls the start of the note and by the interactive object *e* that controls the end of object *sound* and, indirectly, the end of the note C. It is not possible that the interactive point is launched earlier than expected because the temporal relations constrain the possible execution times for all the objects.

3. Timed event structures

Langerak's TESs is a mathematical model to represent systems with non-determinism, real-time and concurrency (Baier, Katoen, and Latella 1998). Henceforth, we call them *event structures*.

Event structures include a set of *labelled events* and a *bundle delay relation* (denoted $\mapsto$). The bundle delay relation establishes which events must happen before some other occurs. Actions can be associated to events. Events are unique, but two events may perform the same action. Events can be defined to be 'urgent'. An *urgent event* occurs as soon as it is enabled. In addition to the bundle relation, event structures include a *conflict relation* establishing events that cannot occur together. Conflict relations are useful, for instance, to model a choice between two parts of a piece or that two parts of the piece are mutually exclusive. Events can also be given *absolute occurrence times*. As an example of an event structure, see Figure 3.

In this paper we present simplified versions of the definitions from Baier, Katoen, and Latella (1998). All events are urgent, there is no conflict relation and absolute occurrence times are in the set $\{0, 1 \ldots\}$. We represent delays as subsets of $\mathbb{N}_0$, as opposed to the original definition of TESs where subsets of the reals is used. Properties of event structures we need in this work, namely the existence of time-consistent event traces, can be easily proved to hold for delays in $\mathbb{N}_0$. All bundle delays are between two events, therefore, we will call them *event delays*. The causality relation is implicitly represented in the event delay function.

DEFINITION 3.1 (Event structure (TES)) *An event structure ε is a tuple $\langle E, l, R, Act\rangle$ with*

- *E, a set of events*,
- $l : E \longrightarrow Act$, *the labelling total function*,
- $R : E \times E \longrightarrow \mathbb{P}(\mathbb{N}_0)$, *the event delay function.*

We denote by the functions $\mathrm{E}(\varepsilon), \mathrm{l}(\varepsilon)$, *and* $\mathrm{R}(\varepsilon)$ *the respective components of* ε *and the set of all* TES*s by* $\mathcal{E}$. *A timed event trace is a sequence* $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ *where* $e_i \in E$ *(all events being pairwise distinct) and time point* $t_i \in \mathbb{N}_0$. *For all* $0 < i, j \leq n$: $e_j \mapsto^{\Delta} e_i \Rightarrow t_i \in t_j + \Delta$.

We denote an event delay $R(e, e') = \Delta$ by $e \overset{\Delta}{\mapsto} e'$ or by $e \mapsto^{\Delta} e'$ and we omit $\{0, 1 \ldots\}$ delays. The *labelling function* for events assigns an element in the set *Act* to each event. The function le : $E \to Act$ returns the label of an event. The *set of all timed event traces* of ε is denoted by

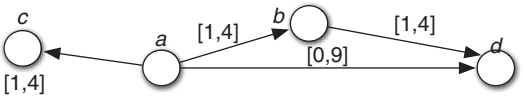

Figure 3. Events *b* and *c* happen between one and four time units after *a*. In addition, event *d* happens between 2 and 8 time units after *a* (the intersection between real-number intervals $[1, 4] + [1, 4]$ and $[0, 9]$).

Traces(ε). Timed event traces comply with causality, but not necessarily the advance of time: This means they can be well-caused but ill-timed. Nonetheless, for any ill-timed trace σ there is always a time-consistent event trace σ' that can be obtained from σ (proved in Baier, Katoen, and Latella 1998). As an example, $\sigma = (a,1), (c,4), (b,3)$ is ill-timed because $(c,4)$ appears before $(b,3)$, and $\sigma' = (a,1), (b,3), (c,4)$ is an equivalent well-timed trace. We only consider well-timed traces in the set Traces(ε).

4. Event structures semantics of IS

We define the formal semantics of IS in event structures. The events represent the start or end points of a TO. An interactive object is represented by a single event. Temporal relations are modelled with event delays. A TO a that is not interactive is represented by an event structure that contains two events se, ee (start and end events resp.) together with all events in the translation of the objects in the subtrees rooted by a. The labels of events are pairs $(type, i)$, where $type \in \{startPoint, endPoint, interactiveObject\}$, and i is the box in the root of the TO giving rise to the event. As an example of the encoding (Figure 4).

DEFINITION 4.1 (Encoding of a TO (eto)) *The encoding of a TO a is a function* $\text{eto} : \mathcal{T} \to \mathcal{E}$. *Let i be the box in the root of a, the encoding of a is defined by*

(1) *if* $i = \langle sp, ep, \{0\}\rangle$ *and a has no subtrees (i.e. a is an interactive object), then*

$$\text{eto}(a) = \langle \{se\}, \{(se, (interactivePoint, i))\}, \emptyset, Act\rangle$$

(2) *if* $i = \langle sp, ep, \Delta \neq \{0\}\rangle$ *(i.e. a is a static TO), then* $\text{eto}(a) = \langle E, l, \emptyset, Act\rangle$, *where*

$$E = \{se, ee\} \cup \bigcup_{x \in \text{subtrees}(a)} \text{E}(\text{eto}(x))$$

$$l = \{(se, (startPoint, i)), (ee, (endPoint, i))\} \cup \bigcup_{x \in \text{subtrees}(a)} \text{l}(\text{eto}(x)).$$

The above definition guarantees that there are unique start and end events in the translation of a static TO, thus we know that each event is related to a single point.

DEFINITION 4.2 (Relation between points and events (*pe*)) *Let o be a TO and P_o the set of points of boxes in o, bijective function* $\text{pe}_o : P_o \to \text{E}(\text{eto(o)})$ *mapping a point $p \in P_o$ to its corresponding event in* $\text{eto}(o)$.

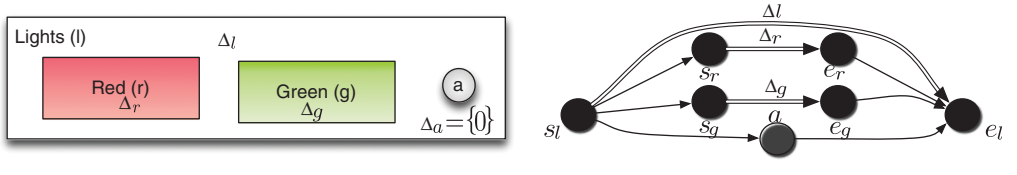

Figure 4. Encoding of a TO and its temporal relations of duration and hierarchy. There are two events for l, two for r, two for g, and a single one for a. Simple arrows starting from s_l and simple arrows arriving to e_l represent the event delays imposed by the hierarchy. Double line arrows are just a visual notation for the event delays that model the duration of the TOs. Start event for an object o is named by s_o and end event by e_o. Interactive object events are named by letters $a, b, c \ldots h$.

Figure 5. Encoding of the TOs r and g, and the Allen's *overlaps* relation between them.

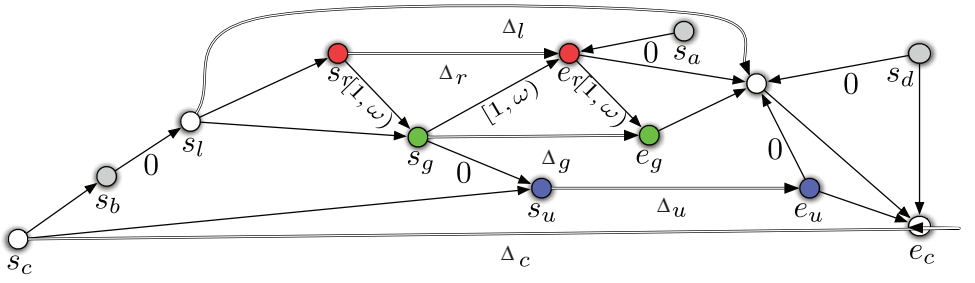

Figure 6. Encoding of the score in Figure 1. Some redundant event delays are removed for simplicity.

DEFINITION 4.3 (Encoding of a temporal relation (etr)) *Each point-to-point relation of an* IS $s = \langle o, R\rangle$ *is represented by an event delay function. The encoding of a temporal relation* r *is given by the function* $\text{etr} : \mathcal{R} \to (E \times E \to \mathbb{P}(\mathbb{N}_0))$. *For each* $r = \langle p, \Delta, q\rangle \in \mathcal{R}$, *the encoding* $\text{etr}(r)$ *is defined by* $\text{pe}_o(p) \overset{\Delta}{\mapsto} \text{pe}_o(q)$.

Figure 5 is the encoding of Allen's relation *r overlaps g*. The encoding of a score is given by adding the event delays from the encoding of the temporal relations to the encoding of the root of the score. The encoding of Figure 1 is presented in Figure 6 and Appendix 2.

DEFINITION 4.4 (Encoding of an IS) *The encoding of an* IS $s = \langle o, R\rangle$ *is given by the function* $\text{es} : \mathcal{S} \to \mathcal{E}$ *that translates* IS *into event structures. Let* $\text{eto}(o) = \langle E, l, \emptyset, Act\rangle$, *then* $\text{es}(s) = \langle E, l, \bigcup_{r\in R} \text{etr}(r), Act\rangle$.

We shall prove that the temporal constraints of the event structures semantics of a score corresponds with the temporal constraints of the score.

DEFINITION 4.5 (Temporal constraint map of an event structure (tc_ε)) *The temporal constraints map of an event structure* ε *is a function* $\text{tc}_\varepsilon : \text{E}(\varepsilon) \to \mathbb{N}_0$ *such that, for each* $e_i \overset{\Delta}{\mapsto} e_j \in \text{R}(\varepsilon)$ *constraint* $\text{tc}_\varepsilon(e_j) \in \text{tc}_\varepsilon(e_i) \oplus \Delta$ *holds.*

Given an event structure ε, it is straightforward to prove that $(e_1, t_1) \ldots (e_n, t_n)$ is a valid trace of ε iff the map $\{(e_1 \mapsto t_1), \ldots, (e_n \mapsto t_n)\}$ is a temporal constraints map of ε. Given two functions f and g, we use the notation $f \circ g$ for the function composition in which first f is applied and then g.

PROPOSITION 4.6 (Equivalence of IS constraints and its event traces) *Let* $s = \langle o, R\rangle$ *be an* IS, $\varepsilon = \text{es}(s)$ *the encoding of the score,* $v : P_o \to \mathbb{N}_0$ *a map and* $\text{pe}_o : P_o \to E$ *the points to events bijective map in the encoding* $\text{es}(s)$. *The function* v *is a viable map of* s *iff* $(pe_o^{-1} \circ v)$ *is a temporal constraints map of* ε.

Proof This proof is presented in detail in Appendix 3. ■

5. Properties

We insist that a motivation of defining an abstract semantics in event structures is to prove properties of the system execution; in particular, properties about the execution traces. The following properties can be defined only with the definition of event structures, TESs and set of traces that we have given in Section 4.

- *Properties of the traces of the execution.*
  - (i) There exists a trace σ that contains a word w. As an example, this property can be used to prove that the sequence of notes C-D-E is part of n traces of execution.
  - (ii) There exists n traces σ that contain a word w, possibly with other events in between; for instance, the sequence of notes C-D-E is contained in one trace $\sigma = (e_C, 0), (e_C, 1), (e_D, 2), (e_C, 3), (e_F, 4), (e_E, 6)$.
  - (iii) The number of possible traces of execution for a score ε is card(Traces(ε)).
  - (iv) If event e is launched before the time unit n, the duration of object a is greater than m. For all $\sigma \in \text{Traces}(\varepsilon)$ and $(e, n), (s_a, t_i), (e_a, t_j) \in \sigma$, it holds that $t_j - t_i \geq m$.
  - (v) After event e is played, there are n traces where event f is launched before event g.
  - (vi) Between time units a to b, there is no more than n objects playing simultaneously.
- *Playability of a score.* This property states that all TOs will be *played* during execution; this is desirable because a score can be over-constrained and therefore not playable. Formally, let $PE_\sigma = \{e | (e, t) \in \sigma\}$ be the events played in trace σ. We say that a score is *playable iff* for all $\sigma \in \text{Traces}(\text{es}(s))$ it holds that that $PE_\sigma = \text{E}(\text{es}(s))$.

  The playability of a score can be decided by solving a *constraint satisfaction problem* (CSP). There exists a $\sigma \in \text{Traces}(\text{es}(s))$ such that $PE_\sigma = \text{E}(\text{es}(s))$ *iff* the following CSP has at least one solution: a variable x_i for each event $e_i \in E$; the domain $\mathbb{N}_F$ for each variable, where $\mathbb{N}_F$ is a finite subset of $\mathbb{N}_0$; and the single constraint tc(ε). This holds as a direct consequence of Proposition 4.6.
- *Minimum duration of a score.* Let s be a score and $\varepsilon = \text{es}(s)$ the encoding of s, the trace whose duration is minimum corresponds to a path from the start event of ε to the end event of ε such that the sum of the delays in the event delay relation is minimal among all paths connecting start and end.
- *Maximum and minimum number of simultaneous TOs.* Let $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ be a trace of $\varepsilon = \text{es}(s)$, and $\text{maxS}(\sigma), \text{minS}(\sigma)$ the maximum and minimum number of events executed simultaneously in σ, respectively. The maximum and minimum number of simultaneous TOs of a score correspond, respectively, to the maximum and minimum value of $\text{maxS}(\sigma)$ and $\text{minS}(\sigma)$ among all $\sigma \in \text{Traces}(\varepsilon)$.

6. Operational semantics of IS

Hitherto, we presented event structures semantics of IS; however, we need operational semantics to model a behaviour that cannot be expressed in the event structures semantics. As an example, we want to express how interactive events launch static events. We chose the `ntcc` calculus to describe operational semantics because it can easily express discrete time, concurrency and temporal constraints.

In Sections 3 and 4, we modelled the durations of the TOs as arbitrary sets of integers. We could express such durations as disjunctions of constraints in `ntcc`; for instance, there is a music improvization problem formalized in `ntcc` by Rueda and Valencia (2004) that uses disjunctions of constraints. Nonetheless, we show in Appendix 4 that the NP-complete *subset sum problem* (Martello and Toth 1990) can be easily translated into the problem of deciding the playability of a

score whose TO durations are arbitrary sets of integers. We argue that the problem of dispatching the events is also NP-complete.

The temporal constraints of an IS can be represented as a *simple temporal problem* (STP) when the durations (of the TOs and the durations between them) are intervals of integers (Dechter, Meiri, and Pearl 1991). The satisfiability of an STP can be easily decided with an algorithm to find *all-pairs shortest-path* of a graph, such as *Floyd–Warshall* (Cormen et al. 2001) which has a polynomial time and space complexity.[1] An STP can be preprocessed to be dispatched online efficiently by relying only on local propagation: looking only to the neighbours of the event launched, as proposed by Muscettola, Morris, and Tsamardinos (1998). In this section, we extend Muscettola et al.'s approach to event structures: we preprocess the temporal constraints of the event structures to be dispatched online efficiently.

In what follows, we introduce `ntcc`, we give an operational semantics of IS based on `ntcc`, and we prove that such a semantics respects the event structures semantics.

6.1. *ntcc calculus*

In *concurrent constraint programming* (`ccp`) (Saraswat 1992), a system is modelled as a collection of concurrent processes whose interaction behaviour is based on the information (represented by constraints) contained in a *global store*. Formally, `ccp` is based on the idea of a constraint system (CS). A CS is composed of a set of (basic) constraints and an entailment relation-specifying constraints that can be deduced from others. In this paper, we use a *finite domain* CS providing arithmetic relations over natural numbers. As an example, using a finite domain CS, we can deduce $pitch \neq 60$ from the constraints $pitch > 40$ and $pitch < 59$.

`Ccp` has a disadvantage: In `ccp` it is not possible to delete nor change information accumulated in the store. For that reason, it is difficult to perceive a notion of discrete time, useful to model reactive systems communicating with an external environment (e.g. sensors and speakers).

`Ntcc` (Nielsen, Palamidessi, and Valencia 2002) circumvents this limitation by introducing the notion of discrete time as a sequence of time units. At each time unit, a `ccp` computation takes place, starting with an empty store (or one that has been given some information by the environment). Concurrent constraints agents operate on this store as in the usual `ccp` model to accumulate information into the store. As opposed to the `ccp` model, the agents can schedule processes to be run in future time units. In addition, since at the beginning of each time unit, a new store is created, information on the value of a variable can change (e.g. it can be forgotten) from one unit to the next. As an example, process **tell** $(pitch_1 = 52) \parallel$ **when** $48 < pitch_1 < 59$ **do next tell** $(instrument = 1)$ adds to the store the constraint $pitch_1 = 52$ and postpones one time unit in the constraint $instrument = 1$. A description of ntcc processes can be found in Table 1.[2] Denotation of processes are sequences of constraints output at each time unit under any possible input.[3]

A simple form of recursion can be defined in terms of basic `ntcc` agents. Recursion can be defined with the form $q(x) \stackrel{\text{def}}{=} P_q$, where q is the process name and P_q is restricted to call q at most once and such a call must be within the scope of a *next* (Valencia 2002, see). The reader should not confuse a *simple definition* with a *recursive definition*; for instance, $Before_{i,j,\Delta} \stackrel{\text{def}}{=}$ **tell**$(P_i + \Delta < P_j)$ is a simple definition where the values of i, Δ and j are replaced 'statically', like a macro in a programming language. On the contrary, a recursive definition such as $Clock(v) \stackrel{\text{def}}{=}$ **tell**$(clock = v) \parallel$ **next** $Clock(v + 1)$ is like a function in a programming language.

[1] There are algorithms to decide the consistency of an STP more efficient than *Floyd–Warshall*.
[2] The agent for asynchrony (*) provided by `ntcc` is omitted in this paper for simplicity.
[3] Operational and denotation semantics are explained in Appendices 7 and 8.

Table 1. Intuitive semantics of the ntcc agents.

Agent	Meaning
tell (c)	Adds c to the current store
when (c) **do** A	If c holds now, run A now
local (x) **in** P	Runs P with local variable x
$A \parallel B$	Parallel composition
next A	Runs A at the next time unit
unless (c) **next** A	Unless c holds now, next run A
$\sum_{i\in I}$ **when** (c_i) **do** P_i	Chooses P_i such that (c_i) holds
$!P$	Executes P each time unit

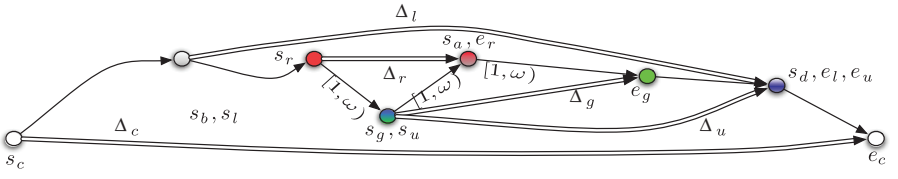

Figure 7. Normal form of the score in Figure 1. Some redundant delays are removed for simplicity.

Figure 8. Normalization rule ($\overset{\text{norm}}{\Longrightarrow} \subseteq \mathcal{E} \times \mathcal{E}$). If there is a zero-duration event delay $a \mapsto^{\{0\}} b$, $\overset{\text{norm}}{\Longrightarrow}$ removes such a delay, combines the labels of a and b, and connects a's predecessors and successors to b.

6.2. *A normal form of the event structures semantics of a score*

An event structure is in normal form if it has no zero-duration event delays. As an example, Figure 7 is the normal form of Figure 1. The normal form collapses simultaneous events into a single event. The normal form simplifies the definition of the operational semantics because we do not have to synchronize two processes to launch two events at the same time. If static and interactive events must happen at the same time and each one is represented as a separated concurrent process, synchronization is difficult because the continuation of an *unless* process must always happen in the next time unit.

To calculate the normal form, we remove sequentially each event delay with zero duration. To remove such a delay, we proceed as shown in Figure 8. To combine labels, we represent the labels of an event structure as sets using function $\text{ls} : (E \times Act) \to (E \times \mathbb{P}(Act))$ defined as $\text{ls}(l) = \{(e, \{label\}) | (e, label) \in l\}$. We denote ε^* an event structure whose labels are sets.

Definition 6.1 (Normal Form of the encoding of a score) *Let $\varepsilon = \langle E, l, R, Act\rangle$ be the encoding of a score, $\varepsilon^* = \langle E, \text{ls}(l), R, Act\rangle$ an event structure whose labels are sets and the relation $\overset{\text{norm}}{\Longrightarrow} \subseteq \mathcal{E} \times \mathcal{E}$ defined in Figure 8. The normal form is obtained by applying $\overset{\text{norm}}{\Longrightarrow}$ until there are no zero-duration delays in the event structure*: $\varepsilon^* \overset{norm^*}{\Longrightarrow} \text{normal}(\varepsilon^*) \not\Longrightarrow$.[4]

The normal form and the encoding of a score do not necessarily have the same event traces; however, they have the same *label traces*. Label traces are similar to time traces, but they relate

[4] Syntax $\overset{\text{norm}^*}{\Longrightarrow}$ means that the $\overset{\text{norm}}{\Longrightarrow}$ rule is applied zero or more times sequentially.

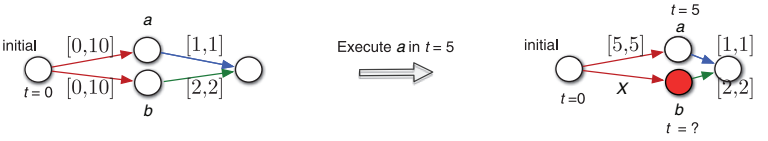

Figure 9. The problem with local propagation. The graph at the left is the constraint graph of an event structure after the *initial* event was executed at $t = 0$ (i.e. the first time unit). At $t = 5$, it seams coherent to execute event a by regarding only the neighbours of a. Such an execution leads to an incoherence because b should have been executed at $t = 4$ to maintain a consistent graph. The solution is to add an event delay labelled by $[1, 1]$ from b to a, which is obtained by transforming into a DES.

each time point to the set of labels that appear in such a time point. Appendix 5 formally defines label traces and shows the correctness of the normal form based on such a definition.

6.3. *Dispatchable event structures*

The normal form does not guarantee that the consistency of the constraints will be maintained during an execution with an algorithm that relies only on local propagation with no search involved. The reader may see this problem in Figure 9. Local propagation is critical for real-time interaction, thus the encoding of the score must be transformed into a *dispatchable event structure* (DES), using an all-pairs shortest-path algorithm such as Floyd–Warshall. This only applies when the duration of the event delays are intervals.

PROPOSITION 6.2 (Correctness of the DESS) *Let ε be an event structure and* $\mathrm{des}(\varepsilon)$ *its* DES *form. Both event structures have equivalent constraints, and in the constraint graph of* $\mathrm{des}(\varepsilon)$, *local propagation simulates full propagation.*

Proof This holds by previous results on STPs (see Appendix 6). ■

6.4. *Ntcc model of a score*

In what follows, we give an `ntcc` model of an IS. We start from the dispatchable normal form of the event structure semantics of the score, namely $\varepsilon^* = \langle E, l^*, R, Act\rangle$. We associate a finite domain variable p_i (event i) with each $e_i \in E$ and we represent every $e_i \mapsto^{\Delta} e_j \in R$ by a constraint $p_j \in p_i + \Delta$. The value ∞, potentially present in some Δ, is considered as a fixed number of the model (denoted n_∞): a given 'big' value in the finite domain CS. We define an `ntcc` process $Delays_R$ posting the above constraints persistently[5]:

$$Delay_{i,j,\Delta} \stackrel{\text{def}}{=} \mathbf{!tell}(p_i + \Delta = p_j) \quad Delays_R \stackrel{\text{def}}{=} \prod_{(e_i,\Delta,e_j)\in R} Delay_{i,j,\Delta}.$$

We have two type of events: interactive (*iEvent*) and static events (*sEvent*). Process $Launch_i$ adds the constraint $launch_i$ in the current time unit, and persistently assigns p_i with the current value of the clock and $launched_i$ to true. User action ac_i is the user event associated with event i: it represents a user interaction. Process *Score* is parametrized by a function $\mathrm{Pr} : P \to \mathbb{P}(P)$ that returns the set of events that are predecessors of a given event, a set $S \subseteq E$, a set $I \subseteq E$ and R.

- Function predecessors Pr is defined by $\mathrm{Pr}(i) = \{j | \langle e_j, e_i, \Delta\rangle \in R\}$.

[5] Agent $\prod$ represents a generalized parallel composition of processes.

- Set I represents the events of interactive objects,

$$I = \{e | e \in E \wedge (\mathit{InteractiveObject}, a) \in \mathrm{le}(e)\}.$$

- Set S represents the events of non-interactive objects,

$$S = \{e | e \in E \wedge (\mathit{InteractiveObject}, a) \notin \mathrm{le}(e)\}.$$

- Process $x \leftarrow y \stackrel{\text{def}}{=} \sum_{v \in \{0,1 \ldots n_\infty\}}$ **when** $v = y$ **do !tell** $(x = y)$ assigns persistently the current value of y to x, and process $\mathit{Eventually}_{\{0,1,\ldots n\}} P \stackrel{\text{def}}{=} P + \mathbf{next} P + \cdots \mathbf{next}^n P$ launches a process P at some time unit within the interval $\{0, 1, \ldots n\}$.

$\mathit{Launch}_i \stackrel{\text{def}}{=}$ **tell**$(\mathit{launch}_i) \parallel p_i \leftarrow \mathit{clock} \parallel$ **!tell** $(\mathit{launched}_i) \parallel$ **next ! tell** $(\neg \mathit{launch}_i)$

$\mathit{iEvent}_{i,\mathrm{Pr}} \stackrel{\text{def}}{=}$ **when** $\bigwedge_{j \in \mathrm{Pr}(i)} \mathit{launched}_j$ **do** (

when $\mathit{clock} + 1 > p_i$ **do unless** $\mathit{launched}_i$ **next** Launch_i

$\parallel$**unless** $\mathit{clock} + 1 < p_i$ **next when** ev_i **do** Launch_i)

$\parallel$**unless** $\mathit{launched}_i$ **next** $\mathit{iEvent}_{i,\mathrm{Pr}}$

$\mathit{sEvent}_{i,\mathrm{Pr}} \stackrel{\text{def}}{=}$ **when** $\bigwedge_{j \in \mathrm{Pr}(i)} \mathit{launched}_j$ **do** (**when** $\mathit{clock} + 1 < p_i$ **do next** $\mathit{sEvent}_{i,\mathrm{Pr}}$

$\parallel$ **unless** $\mathit{clock} + 1 < p_i \vee \mathit{launched}_i$ **next** Launch_i)

$\parallel$**unless** $\mathit{launched}_i$ **next** $\mathit{sEvent}_{i,\mathrm{Pr}}$

$\mathit{Events}_{I,S,\mathrm{Pr}} \stackrel{\text{def}}{=} \prod_{e_i \in I} \mathit{iEvent}_{i,\mathrm{Pr}} \parallel \prod_{e_i \in S} \mathit{sEvent}_{i,\mathrm{Pr}}$

- Process User chooses between launching or not a user action. Clock ticks forever. Process Score is parametrized by I, S, Pr and R. Score adds a constraint $0 < p_i < n_\infty$ to allow that $\mathit{clock} + 1 < p_i$ be eventually deduced even when an object's duration is said to be infinite.

$$\mathit{User}_I \stackrel{\text{def}}{=} \left(\prod_{e_i \in I} (\mathit{Eventually}_{\{0,1 \ldots n_\infty\}} \mathbf{tell}(ac_i)) + \mathbf{skip} \right)$$

$$\mathit{Clock}(k) \stackrel{\text{def}}{=} \mathbf{when}\ k < n_\infty - 1\ \mathbf{do}\ (\mathbf{tell}\ (\mathit{clock} = k) \parallel \mathbf{next} \mathit{Clock}(k+1))$$

$$\mathit{Score}_{I,S,\mathrm{Pr},R} \stackrel{\text{def}}{=} \mathit{Events}_{I,S,\mathrm{Pr}} \parallel \mathit{Delays}_R \parallel \mathit{User}_I \parallel \mathit{Clock}(0) \parallel \prod_{i \in S \cup I} \mathbf{!tell}(0 < p_i < n_\infty).$$

We shall prove that the `ntcc` model is correct with respect to the dispatchable normal form of the event structures semantics of a score; therefore, any possible output of `ntcc` respects the temporal constraints of the event structures semantics of the score.

PROPOSITION 6.3 *Let s be a score, $\varepsilon^* = \mathrm{des}(\mathrm{normal}(\mathrm{es}(s))) = \langle E, l^*, R, Act \rangle$ its event structures semantics, $\mathrm{tc}(\varepsilon^*)$ its temporal constraints, $P = \mathit{Score}_{S,I,\mathrm{Pr},R}$ the `ntcc` process that represents the score, $[[P]]$ the denotation of the `ntcc` process, and T_i is the set of indexes j such that $[[P]]_{j+1}$ entails launch_i. It holds for all sequences in $[[P]]$, $n = \mathrm{card}(E)$, that $T_1 \times T_2 \cdots \times T_n \subseteq \mathrm{Solutions}(\mathrm{tc}(\varepsilon^*))$.*

Proof Proof is presented in detail in Appendix 9. ■

7. Conclusions and future work

Most scenarios and musical pieces with interactive controls have no formal semantics. IS is a formalism to describe interactive scenarios based on temporal constraints. In this paper, we introduced an event structures semantics of IS, we formalized some properties, and we proved that the event structures semantics complies with the temporal constraints of the score. With the event structures semantics, we expressed several properties about the traces of execution that are difficult to express and prove using constraints.

We introduced the DESS: event structures whose TO durations and temporal distances among objects are integer intervals. DESS can be dispatched online by relying only on local propagation. This is achieved by transforming the constraint graph into an *all-pairs shortest-path graph*; however, that drastically increases the number of arcs. In the future, we propose to minimize such networks as proposed by Muscettola, Morris, and Tsamardinos (1998).

Although event structures provide a theoretical background to specify properties and understand the system, there is no difference between interactive objects and non-interactive TOs in the event structures semantics: such a difference can only be expressed in the operational semantics. We presented an operational semantics based on the dispatchable normal form of the event structures of the score. A score is in normal form when it does not have zero-duration event delays. The computation of the normal form is similar to the algorithm to transform a score into a Petri net proposed by Allombert (2009): In Petri nets semantics of IS, points of TOs executed at the same time share the same *place* (i.e. state).

We believe that this paper extends significantly Allombert's model because it provides a concise operational semantics for IS whose TO duration can be any interval of integers. Temporal relations with flexible integer intervals such as $\{0\}$, $\{0, 1, \ldots\}$ and $\{1, 2, \ldots\}$ are proposed by Allombert et al. (2006) and Allombert, Assayag, and Desainte-Catherine (2008). In fact, arbitrary integer intervals are not allowed in neither Virage nor iScore. To handle temporal relations with arbitrary intervals, Allombert (2009) proposed to either build a *Hierarchical coloured time stream Petri net*, adding a big number of new places (states), or to use a constraint store that is unrelated to the Petri nets semantics, and the combined semantics of Petri nets interacting with a constraint store are not given. Note that a conditional branching extension is also presented in Allombert (2009).

Event structures semantics could be extended to express conditional branching. Using time event structures with conflicts, it is possible to model conditional branching (i.e. the possibility to choose among different continuations of the piece based on the preferences of the musician). In addition, Langerak (1992) describes how to encode recursive processes into event structures; in fact, loops in IS could be encoded with such a technique. Unfortunately, conditional branching drastically increases the complexity of the system; for instance, a score may contain dead-locks. We must aim for automated verification. An alternative for automated verification is *constraint programming*; for instance, to verify the playability of a score and calculate the potential time positions of the points of the score. Nonetheless, for some properties, the notion of *trace*, is more appropriate.

There is another possible extension to our model. The model of Allombert et al. includes a *nominal duration* and a *nominal starting time* for the TOs. A key concept in interactive music scores is the nominal duration of a TO: a value preferred by the composer, which may change due to the interactive objects. Music scores must have a nominal duration instead of non-deterministic or probabilistic durations. The nominal duration in our operational semantics is the minimum duration of a TO; the nominal starting time is also the minimum value allowed. A future direction is to provide user-defined nominal values.

Another advantage of our event structure semantics and our operational semantics is that they can express *trans-hierarchical relations*: temporal relations between objects with different parents. Trans-hierarchical relations are not possible to model with hierarchical time stream Petri nets used by Allombert et al. This is useful, for instance, to model temporal relations between videos and sounds that are contained in different TOs.

Acknowledgements

The authors are grateful to the reviewers for their comments.

References

Allen, James F. 1983. "Maintaining Knowledge about Temporal Intervals." *Communication of ACM* 26 (11), 832–843.

Allombert, Antoine. 2009. "Aspects temporels d'un système de partitions numèriques interactives pour la composition et l'interprétation." PhD in Computer science, Université de Bordeaux. 351, cours de la Liberation, Talence.

Allombert, Antoine, Gérard Assayag, M. Desainte-Catherine, and Camilo Rueda. 2006. "Concurrent Constraint Models for Interactive Scores." In *Proc. of SMC '06,* Marseille, France: European Commission.

Allombert, Antoine, Gérard Assayag Assayag, and Myriam Desainte-Catherine. 2008. "Iscore: A System for Writing Interaction." In *Proc. of DIMEA '08,* 360–367. New York: ACM.

Allombert, Antoine, Myriam Desainte-Catherine, and Gerard Assayag. 2008. "De Boxes à Iscore: vers une écriture de l'interaction." In *Proc. of JIM 2008*. Albi, France: Association Française d'Informatique Musicale.

Allombert, Antoine, Pascal Baltazar, Raphaël Marczak, Myriam Desainte-Catherine, and Laurent Garnier. 2010. "Designing an Interactive Intermedia Sequencer from Users Requirements and Theoretical Background." In *Proc. of ICMC 2010*. New York: International Computer Music Association.

Baier, Christel, Joost-Pieter Katoen, and Diego Latella. 1998. "Metric Semantics for True Concurrent Real Time." In *Proc. of ICALP '98*. Berlin, Germany: Springer.

Bresson, Jean, Carlos Agón, and Gérard Assayag. 2005. "OpenMusic 5: A Cross-Platform Release of the Computer-Assisted Composition Environment." In *10th Brazilian Symposium on Computer Music*. Rio de Janeiro, Brazil: Brazilian Computing Society.

Cormen, Thomas H., Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. 2001. *Introduction to Algorithms*. Chap. 5, 2nd ed. New York City: McGraw-Hill Higher Education.

Dechter, Rina, Itay Meiri, and Judea Pearl. 1991. "Temporal Constraint Networks." *Artificial Intelligence* 49 (1–3): 61–95.

Desainte-Catherine, M., and N. Brousse. 2003. "Towards a Specification of Musical Interactive Pieces." In *Proc. of CIM XIX*. Firenze, Italy: AIMI – Associazione di Informatica Musicale Italiana.

Gennari, Rosella. 1998. "Temporal Resoning and Constraint Programming – A Survey." *CWI Quaterly* 11 (16): 3–163.

Langerak, Rom. 1992. "Bundle Event Structures: A Non-Interleaving Semantics for LOTOS." In *Formal Description Techniques, V, Proceedings of the IFIP TC6/WG6.1 Fifth International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols, FORTE 92, Perros-Guirec, France, 13–16 October 1992,* Vol. C-10 of *IFIP Transactions*331–346. Twente, Holland: North-Holland.

Martello, Silvano, and Paolo Toth. 1990. *Knapsack Problems: Algorithms and Computer Implementations*. Chap. 1. New York: John Wiley & Sons, Inc.

Muscettola, Nicola, Paul H. Morris, and Ioannis Tsamardinos. 1998. "Reformulating Temporal Plans for Efficient Execution." In *Proc. of Principles of Knowledge Representation and Reasoning,* 444–452. Trento, Italy: AAAI Digital Library.

Nielsen, M., C. Palamidessi, and F. Valencia. 2002. "Temporal Concurrent Constraint Programming: Denotation, Logic and Applications." *Nordic Journal of Computing* 1 (9): 145–188.

Olarte, Carlos, and Camilo Rueda. 2009. "A Declarative Language for Dynamic Multimedia Interaction Systems." In *Proc. of Mathematics and Computation in Music,* Vol. 38. Berlin, Germany: Springer.

Planken, Léon, Mathijs de Weerdt, and Neil Yorke-Smith. 2010. "Incrementally Solving STNs by Enforcing Partial Path Consistency." In *ICALPS,* 129–136. Bordeaux, France: Springer LNCS (International).

Puckette, Miller, Theodore Apel, and David Zicarelli. 1998. "Real-Time Audio Analysis Tools for Pd and MSP." In *Proc. of ICMC '98*. Ann Arbor: International Computer Music Association.

Rueda, C., and F. Valencia. 2004. "On Validity in Modelization of Musical Problems by CCP." *Soft Computing* 8 (9): 641–648.

Saraswat, Vijay A. 1992. *Concurrent Constraint Programming*. Cambridge, MA: MIT Press.

Toro, Mauricio, Carlos Agón, Gérard Assayag, and Camilo Rueda. 2009. "Ntccrt: A Concurrent Constraint Framework for Real-Time Interaction." In *Proc. of ICMC '09*. Montreal, Canada: International Computer Music Association.

Toro, Mauricio, and Myriam Desainte-Catherine. 2010. "Concurrent Constraint Conditional Branching Interactive Scores." In *Proc. of SMC '10*. Barcelona, Spain: European Commission.

Valencia, Frank D. 2002. "Temporal Concurrent Constraint Programming." PhD in Computer Science, University of Aarhus. Aarhus, Denmark.

Valencia, Frank D. 2005. "Decidability of Infinite-State Timed CCP Processes and First-Order LTL." *Theoretical Computer Science – Expressiveness in Concurrency* 330 (3): 557–607.
Vickery, Lindsay. 2004. "Interactive Control of Higher Order Musical Structures." In *Proc. of ACMC*. Victoria University, New Zealand: ACMA.
Xu, Lin, and Berthe Y. Choueiry. 2003. "A New Efficient Algorithm for Solving the Simple Temporal Problem." In *Proceedings of the 10th International Symposium on Temporal Representation and Reasoning/4th International Conference on Temporal Logic* 212–222. Los Alamitos, CA: IEEE Computer Society.

Appendix 1. Structural definitions and temporal constraints of Figure 1

Table A1 presents the structure of the score in Figure 1 and Table A2 its temporal constraints. The Allen's relation *r overlaps g* is represented by the three last point-to-point relations in the Table A2.

Table A1. Structural definition of the score in Figure 1.

Points and TOs	Explicit temporal relations
$I = \{c, l, r, g, u, a, b, d\}$	$R = \{$
$H = \{(c,l),(c,u),(c,b),(c,d),(l,r),(l,g),(l,a)\}$	$\langle ep_b, \{0\}, sp_l\rangle$,
$P = \{sp_c, sp_l, sp_r, sp_g, sp_u, sp_a, sp_b, sp_d,$	$\langle sp_g, \{0\}, sp_u\rangle$,
$ep_c, ep_l, ep_r, ep_g, ep_u, ep_a, ep_b, ep_d\}$	$\langle ep_a, \{0\}, ep_r\rangle$,
$c = \langle sp_c, ep_c, \Delta_c\rangle$	$\langle ep_u, \{0\}, ep_l\rangle$,
$l = \langle sp_l, ep_l, \Delta_l\rangle$	$\langle ep_d, \{0\}, ep_l\rangle$,
$r = \langle sp_r, ep_r, \Delta_r\rangle$	$\langle sp_r, (0,\infty), sp_g\rangle$,
$g = \langle sp_g, ep_g, \Delta_g\rangle$	$\langle ep_r, (0,\infty), ep_g\rangle$,
$u = \langle sp_u, ep_u, \Delta_u\rangle$	$\langle sp_g, (0,\infty), ep_r\rangle$
$a = \langle sp_a, ep_a, \{0\}\rangle$	$\}$
$b = \langle sp_b, ep_b, \{0\}\rangle$	
$d = \langle sp_d, ep_d, \{0\}\rangle$	

Note: A TO is a directed tree $o = \langle I, H\rangle$ and a score is a tuple $\langle o, R\rangle$.

Table A2. Implicit and explicit temporal constraints of the score in Figure 1.

Durations	Hierarchy	Ex. temporal relations
$\nu(\text{ep}(c)) \in \nu(\text{sp}(c)) + \text{d}(c)$	$\nu(\text{sp}(l)) \geq \nu(\text{sp}(c)) \wedge \nu(\text{ep}(c)) \geq \nu(\text{ep}(l))$	$\nu(\text{ep}(b)) = \nu(\text{sp}(l))$
$\nu(\text{ep}(l)) \in \nu(\text{sp}(l)) + \text{d}(l)$	$\nu(\text{sp}(u)) \geq \nu(\text{sp}(c)) \wedge \nu(\text{ep}(c)) \geq \nu(\text{ep}(u))$	$\nu(\text{ep}(d)) = \nu(\text{ep}(l))$
$\nu(\text{ep}(r)) \in \nu(\text{sp}(r)) + \text{d}(e)$	$\nu(\text{sp}(b)) \geq \nu(\text{sp}(c)) \wedge \nu(\text{ep}(c)) \geq \nu(\text{ep}(b))$	$\nu(\text{sp}(g)) = \nu(\text{sp}(u))$
$\nu(\text{ep}(g)) \in \nu(\text{sp}(g)) + \text{d}(g)$	$\nu(\text{sp}(d)) \geq \nu(\text{sp}(c)) \wedge \nu(\text{ep}(c)) \geq \nu(\text{ep}(d))$	$\nu(\text{ep}(a)) = \nu(\text{ep}(r))$
$\nu(\text{ep}(u)) \in \nu(\text{sp}(u)) + \text{d}(u)$	$\nu(\text{sp}(r)) \geq \nu(\text{sp}(l)) \wedge \nu(\text{ep}(l)) \geq \nu(\text{ep}(r))$	$\nu(\text{ep}(u)) = \nu(\text{ep}(l))$
$\nu(\text{ep}(a)) \in \nu(\text{sp}(a)) + \{0\}$	$\nu(\text{sp}(a)) \geq \nu(\text{sp}(l)) \wedge \nu(\text{ep}(l)) \geq \nu(\text{ep}(a))$	$\nu(\text{sp}(g)) > \nu(\text{sp}(r))$
$\nu(\text{ep}(b)) \in \nu(\text{sp}(b)) + \{0\}$	$\nu(\text{sp}(g)) \geq \nu(\text{sp}(l)) \wedge \nu(\text{ep}(l)) \geq \nu(\text{ep}(g))$	$\nu(\text{ep}(g)) > \nu(\text{ep}(r))$
$\nu(\text{ep}(d)) \in \nu(\text{sp}(d)) + \{0\}$		$\nu(\text{sp}(g)) < \nu(\text{ep}(r))$

Note: Interval $\{0, 1, \ldots\}$ is represented by $\geq$, set $\{0\}$ by $=$, and interval $\{1, 2, \ldots\}$ by $<$.

Appendix 2. Example of the event structures semantics of a score

As an example, the encoding of Figure 1 is presented as follows. The event delay relation is presented in three subsets: derived from the durations, from the hierarchy and from the explicit temporal relations. Figure 6 is a visual representation of the encoding. Let $\varepsilon = \langle E, l, R, Act\rangle$ such that

- $$E = \{s_c, e_c, s_l, e_l, s_r, e_r, s_g, e_g, s_u, e_u, s_a, s_b, s_d\}$$

•

$$\begin{aligned} l = &\{(s_c, (startPoint, c)), (e_c, (endPoint, c)), (s_l, (startPoint, l)), (e_l, (endPoint, l)),\\ &(s_r, (startPoint, r)), (e_r, (endPoint, r)), (s_g, (startPoint, g)),\\ &(e_g, (endPoint, g)), (s_u, (startPoint, u)), (e_u, (endPoint, u)),\\ &(s_a, (interactiveObject, a)), (s_b, (interactiveObject, b)), (s_d, (interactiveObject, d))\} \end{aligned}$$

•

$$\begin{aligned} R = &\{s_c \mapsto^{\Delta_c} e_c, s_l \mapsto^{\Delta_l} e_l, s_r \mapsto^{\Delta_r} e_r, s_g \mapsto^{\Delta_g} e_g, s_u \mapsto^{\Delta_u} e_u\}\\ &\cup \{s_c \mapsto s_b, s_c \mapsto s_u, s_l \mapsto s_r, s_l \mapsto s_g, e_l \mapsto e_c, e_u \mapsto e_c, e_d \mapsto e_c, e_g \mapsto e_l, e_r \mapsto e_l\}\\ &\cup \{s_r \mapsto^{(0,\infty)} s_g, s_g \mapsto^{(0,\infty)} e_r, e_r \mapsto^{(0,\infty)} e_g, s_b \mapsto^{\{0\}} s_l, s_d \mapsto^{\{0\}} e_l, s_a \mapsto^{\{0\}} e_r, e_u \mapsto^{\{0\}} e_l, s_g \mapsto^{\{0\}} s_u\}. \end{aligned}$$

Appendix 3. Proof of Proposition 4.6

We recall that $\nu : P \to \mathbb{N}$ gives the *time positions* for each point $p \in P$. We also recall the notation for temporal constraints: $t \oplus \Delta = \{t' | t' = t + \delta, \delta \in \Delta\}$.

Proof Let $s = \langle o, R\rangle$ be a score, $\langle p, r, q\rangle \in R$ a temporal relation, $p, q \in P_o$, p are points of boxes in o. If ν is viable, we have $\nu(q) \in \nu(p) \oplus \Delta$; by Definition 4.3, $\text{etr}(\langle p, \Delta, q\rangle) = \text{pe}(p) \mapsto^{\Delta} \text{pe}(q)$. Let $pe(p) = e_1$ and $pe(q) = e_2$. by Definition 4.5, a temporal constraints map tc for $ts(s)$ must obey $tc(e_2) \in tc(e_1) \oplus \Delta$. It is straightforward to show that this is the case when $\text{tc} = pe^{-1} \circ \nu$ provided ν is viable. ■

Appendix 4. Time complexity of the playability of a score

In what follows, we will show that that deciding the playability of a score is NP-complete in the general case, but there is an interesting subclass that is tractable.

A.1. *The NP-complete case*

We will show that the decision problem of the *subset sum* (Martello and Toth 1990) can be encoded as the playability of an IS. The subset problem is stated as follows: Given a set of integers $\{w_1, w_2 \ldots w_n\}$ of n objects and an integer W, does any non-empty subset sum to W?

$$\sum_{i=1}^{n} w_i.e_i = W, e_i \in \{0, 1\}, \quad \text{where at least one } x_i \neq 0. \tag{1}$$

There are several algorithms to solve the subset sum, all with exponential time complexity in n, the number of objects. The most naïve algorithm would be to cycle through all subsets of k numbers and, for every one of them, check if the subset sums to the right number. The running time is of the order $O(n2^n)$, since there are 2^n subsets and, to check each subset, we need to sum at most n elements. The best algorithm known runs in time $O(2^{n/2})$ (Martello and Toth 1990).

In what follows, we present the temporal constraints of the score shown in Figure A1.[6] For $1 \leq i \leq n$, $e_i \in s_i + \{0, w_i\}$, $e_s = s_s + W$, $e_i = e_s$, and $s_s = s_i$. The reader can easily verify that these constraints are equivalent to the constraint in Equation (1).

It is trivial to show that there is a polynomial algorithm to check whether a solution satisfies the problem. Such an algorithm replaces the values of e_i in Equation (1). In conclusion, the subset sum can be encoded as the playability (i.e. satisfiability) of an IS and there is a polynomial-time algorithm to check if the solution is true, thus the satisfiability of an IS is NP-complete.

A.2. *A polynomial-time subclass*

The temporal constraints of an IS can be represented as an STP when the domains of the durations are intervals of integers without holes (Dechter, Meiri, and Pearl 1991). An STP consists of a set of point variables $X = \{x_1, \ldots x_n\}$ and binary constraints over those variables $C = \{c_1, \ldots, c_n\}$ of the form $c_k : x_i - x_j \in [a, b]$ with $x_i, x_j \in X$ and $a, b \in \mathbb{N}_0$.

[6] Knapsack picture is taken from the Wikipedia. http://en.wikipedia.org/wiki/Knapsack_problem

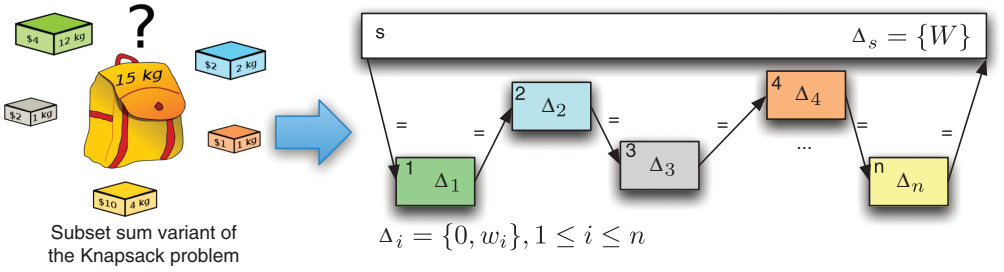

Figure A1. Encoding of the subset sum problem into an IS. Note that the subset sum problem is a variant of the knapsack decision problem where costs are not taken into account and the goal is to find if there is a subset of the elements that fills exactly the knapsack capacity.

The satisfiability of an STP can be easily computed with an algorithm to find *the all-pairs shortest-path* of a graph, such as the *Floyd–Warshall* (Cormen et al. 2001) algorithm which has a polynomial time and space complexity.[7] There are faster algorithms for this problem in the literature (Planken, de Weerdt, and Yorke-Smith 2010; Xu and Choueiry 2003); however, they are efficient to calculate if an STP has a solution, but does not guarantee that the constraint problem remains satisfiable when dispatching the events during the execution of a score.

Appendix 5. Correctness of the normal form

We recall that we use the notation $\varepsilon^* = \langle E, l^*, R, Act\rangle$ to represent an event structure whose labels are sets.

DEFINITION A.1 (Label trace) *Given an event structure* $\varepsilon^* = \langle E, l^*, R, Act\rangle$ *and* $\sigma \in \text{Traces}(\varepsilon^*)$, *a label trace is a sequence* $\sigma_L = (L_1, t_1)\ldots(L_n, t_n)$ *with* $L_i \in \{y | y \in \text{range}(\text{l}(\varepsilon))\}, t_i \in (\mathbb{N} \cup \{\infty\})$ *(all label sets being pairwise disjoint and all time points being pairwise different), such that for all* $1 \leq i \leq n$, *there is an element* $(e, t_i) \in \sigma$ *with* $\text{le}(e) \subseteq L_i$ *and* $(L_i, t_i) \in \sigma_L$. *We denote* $\text{l-Traces}(\varepsilon)$ *the set of all label traces of* ε.

We say that two event structures $\varepsilon^*, \varepsilon'^*$ have the same label traces *iff* for each $(L, t) \in \sigma, \sigma \in \text{l-Traces}(\varepsilon^*)$ and $(L', t') \in \sigma', \sigma' \in \text{l-Traces}(\varepsilon'^*)$, if $t = t'$ then $L = L'$.

PROPOSITION A.2 *An event structure* $\varepsilon = \langle E, l, R, Act\rangle$ *and its normal form have the same label traces. Let* $\varepsilon^* = \langle E, \text{ls}(l), R, Act\rangle$. *It holds that* $\text{l-Traces}(\varepsilon^*) = \text{l-Traces}(\text{normal}(\varepsilon^*))$.

Proof We must prove that there is a finite sequence of derivations from ε^* to $\text{normal}(\varepsilon^*)$ applying the norm rule. Let $\varepsilon^* = \langle E, l^*, R\rangle$ and $\text{normal}(\varepsilon^*) = \langle E', l'^*, R'\rangle$. We will proceed by induction over the sequence $\varepsilon^* \overset{\text{norm}}{\Longrightarrow} \varepsilon'^* \overset{\text{norm}^*}{\Longrightarrow} \text{normal}(\varepsilon) \not\Longrightarrow$

(1) *Base case.* Let $f, g \in \mathcal{E}$ be the event structures in the left and right side of Figure 8, respectively. First we have to prove that $\text{l-Traces}(f) = \text{l-Traces}(g)$. Events a and b happen at the same time point in f, then labels L_a and L_b occur at the same time in the label traces of f. Event b occurs at the same time in g and f because b has the successors and predecessors of a in g, therefore, labels L_a and L_b occur at the same time point in the label traces of both f and g. Since L_a and L_b occurs at the same time in both f and g, for any $a \in \text{E}(\varepsilon^*), b \in \text{E}(\varepsilon^*)$ such that $a \mapsto^{\{0\}} b$, it holds that $\text{l-Traces}(\varepsilon^*) = \text{l-Traces}(\varepsilon'^*)$.

(2) *Inductive case.* Induction over the sequence of derivations $\varepsilon^* \overset{\text{norm}}{\Longrightarrow} \varepsilon'^* \overset{\text{norm}^*}{\Longrightarrow} \text{normal}(\varepsilon^*) \not\Longrightarrow$. The argument follows as for the base case because there is only one rule. The sequence of derivations is finite because the normalization rule always reduces the number of events and event delays. ■

Appendix 6. Proof of Proposition 6.2

Proof This holds by previous results on STPs. Theorems 'every *all-pairs shortest-path network* (APSPN) is dispatchable [using local propagation only]', presented in Muscettola, Morris, and Tsamardinos (1998); locally consistent assignment can be extended to a global one in a dispatchable graph, presented in Muscettola, Morris, and Tsamardinos (1998); and equivalence between an APSPN and the original network, presented in Dechter, Meiri, and Pearl (1991). ■

[7] Floyd–Warshall has a time complexity of $O(n^3)$, where n is the number of events of the event structures semantics of the score.

Table A3. Operational semantics of ntcc.

TELL	$\dfrac{}{(\mathbf{tell}(c), d) \longrightarrow (\mathbf{skip}, d \wedge c)}$	PAR	$\dfrac{(P, c) \longrightarrow (P', d)}{(P \| Q, c) \longrightarrow (P' \| Q, d)}$
REP	$\dfrac{}{(!P, d) \longrightarrow (P \| \mathbf{next}\ !P, d)}$	STR	$\dfrac{\gamma_1 \longrightarrow \gamma_2}{\gamma_1' \longrightarrow \gamma_2'}$ **if** $\gamma_1 \equiv \gamma_1'$ and $\gamma_2 \equiv \gamma_2'$
SUM	$\dfrac{}{(\sum_{i \in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i, d) \longrightarrow (P_j, d)}$ **if** $d \models c_j, j \in I$		
UNL	$\dfrac{}{(\mathbf{unless}\ c\ \mathbf{next}\ P, d) \longrightarrow (\mathbf{skip}, d)}$ **if** $d \models c$		
LOC	$\dfrac{(P, c \wedge \exists_x d) \longrightarrow (P', c')}{((\mathbf{local}\ x, c)P, d) \longrightarrow ((\mathbf{local}\ x, c')P', d \wedge \exists_x c')}$		
OBS	$\dfrac{(P, c) \longrightarrow^* (Q, d) \not\longrightarrow}{P \overset{(c,d)}{\Longrightarrow} R}$ **if** $R \equiv F(Q)$		

Appendix 7. Operational Semantics of ntcc

Ntcc has two type of transitions: internal transitions that occur within a time unit ($\longrightarrow$) and observable transitions that occur from one time unit to the other ($\Longrightarrow$). A rule states that whenever the conditions have been obtained in the course of some derivation, the conclusion is also obtained (see Table A3).

Rule PAR states that whenever a process P can evolve into a process P' by performing an internal transition, it can also evolve from $P \| Q$ to $P' \| Q$. Rules TELL, SUM and UNL are interpreted in a similar way to rule PAR. Rule REP specifies that process $!P$ produces a copy of P at the current time unit, and then persists in the next time unit. Rule STR says that structurally congruent (defined in detail in Nielsen, Palamidessi, and Valencia 2002) configurations have the same reductions. Rule LOC explains how x can have a local scope in the sense of local variables in programming languages.

Rule OBS says that an observable transition from P labelled by (c, d) is obtained by performing a sequence of internal transitions from the initial configuration (P, c) to a final configuration (Q, d) in which no further internal evolution is possible.[8] The residual process R to be executed in the next time unit is equivalent to the process $F(Q)$, where the future function $F : Proc \to Proc$ is a the partial function defined by

$$F(Q) = \begin{cases} \mathbf{skip} & \text{if } Q = \sum_{i \in I} \mathbf{when}\ c_i\ \mathbf{do}\ Q_i \\ F(Q_1) \| F(Q_2) & \text{if } Q = Q_1 \| Q_2 \\ (\mathbf{local}\ x) F(R) & \text{if } Q = (\mathbf{local}\ x, c) R \\ R & \textit{if } Q = \mathbf{next}\ R \textit{ or } Q = \mathbf{unless}\ c\ \mathbf{next}\ R \end{cases}$$

The process $F(Q)$, defined above, is obtained by removing from Q summations that did not trigger activity within the current time unit and any local information which has been stored in Q. Process *skip* is a short form for an empty summation: a process that does not perform any transition.

Appendix 8. Denotational semantics of ntcc

The denotation of an ntcc process is a set of infinite sequences of stores (i.e. constraints). Each element of the sequence corresponds to each time unit. A function [.] associates such sets to each process (see Table A4). Notation $\exists_X \alpha$ denotes the application of $\exists_X$ to each constraint on α. Set C is the set of constraints in the cs. Henceforth C^ω, C^* are the set of infinite and finite sequences of constraints in C, respectively. We use α, α' to represent elements of C^ω, β to represent an element of C^*, and $\beta.\alpha$ to represent the concatenation of β and α.

Intuitively, $[\![P]\!]$ represents the sequences that a process P can output interacting with any possible environment. As an example, the sequences of $[\![\mathbf{tell}(c)]\!]$ are those whose first element is stronger than c (rule $D1$). Further information on the denotation of ntcc can be found in Nielsen, Palamidessi, and Valencia (2002).

The infinite sequences of the denotation of a process can be represented as a Büchi automaton. Büchi automata are an extension of finite state automata for infinite input. Such an automata accepts exactly those runs in which at least one of the infinitely often occurring states is an accepting state. For simplicity, we label the transitions as $c \vdash d$ (i.e. there is a transition for every constraint $c \in C_{finite}$ stronger than d). The reader may find the encoding of the denotation of ntcc into Büchi in Valencia (2005). The encoding also defines a finite set of constraints (i.e. *the relevant constraints of P*) because the alphabet of a Büchi automaton must be finite.

[8] We use c, d in this paper to represent constraint stores.

Table A4. Denotational semantics of ntcc.

D1	$[\![\mathbf{tell}(c)]\!] = \{d.\alpha \mid d \vdash c, \alpha \in C^\omega\}$
D2	$[\![\sum_{i\in I} \mathbf{when}\ c_i\ \mathbf{do}\ P_i]\!] = \bigcup_{i\in I}\{d.\alpha \mid d \vdash c_i, d.\alpha \in [\![P_i]\!]\} \cup \bigcap_{i\in I}\{d.\alpha \mid d \nvdash c_i, d.\alpha \in C^\omega\}$
D3	$[\![P \| Q]\!] = [\![P]\!] \cap [\![Q]\!]$
D4	$[\![\mathbf{local}\ x\ \mathbf{in}\ P]\!] = \{\alpha \mid$ there exists $\alpha' \in [\![P]\!]$ s.t. $\exists_x \alpha = \exists_x \alpha'\}$
D5	$[\![\mathbf{next}\ P]\!] = \{d.\alpha \mid d \in C, \alpha \in [\![P]\!]\}$
D6	$[\![\mathbf{unless}\ c\ \mathbf{next}\ P]\!] = \{d.\alpha \mid d \vdash c, \alpha \in C^\omega\} \cup \{d.\alpha \mid d \nvdash c, \alpha \in [\![P]\!]\}$
D7	$[\![!P]\!] = \{\alpha \mid \forall \beta \in C^*, \alpha' \in C^\omega s.t. \alpha = \beta.\alpha'$, we have $\alpha' \in [\![P]\!]\}$

Appendix 9. Proof of Proposition 6.3

Proof We shall proceed by induction over the structure of ε^*. We prove the proposition by contradiction. We suppose that there is at least a tuple in $T_1 \times T_2 \cdots \times T_n$ that is not a solution of $\mathrm{tc}(\varepsilon^*)$. In what follows, we do not consider the process $User_i$ in the proof because its denotation is the set of all the possible sequences of stores (i.e. constraints). The constraint $launch_i$ can only be deduced in a single time unit because after such a time unit, the constraint $\neg launch_i$ is added in all subsequent time units.

(1) *A single interactive object.* The encoding of a score with a single interactive object has one event e_i, no boxes, and $\mathrm{tc}(\varepsilon^*) = t_i \in \mathbb{N}$.

Figure A2 presents the denotation of $iEvent_{i,\mathrm{Pr}}$ and Figure A4 the denotation of $Clock(0)$. In what follows, we analyse $[\![Score_{I,S,\mathrm{Pr},R}]\!]$, which is the intersection of the denotations of $iEvent_{i,\mathrm{Pr}}$, $Clock(0)$ and $!\mathbf{tell}(0 < p_i \leq n_\infty)$. Since e_i has no predecessors, the constraint $\bigwedge_{j\in \mathrm{Pr}}\ launched_j$ can always be deduced. Note that whenever ev can be deduced (after the first time unit), $launch_i$ can also be deduced. Finally, $clock + 1 > p_i$ can be deduced when $clock = n_\infty - 1$ because $0 < p_i < n_\infty$ can always be deduced, thus if $clock + 1 > p_i$ can be deduced, $launch_i$ can be deduced in the position n_∞. Therefore, $T_i = [1, n_\infty]$

(2) *Two events.* The encoding of a score with one TO has two events e_i and e_j, an event delay $e_i \mapsto^{\Delta} e_j$ and $\mathrm{tc}(\varepsilon^*) = t_i \in \mathbb{N} \wedge t_j \in \mathbb{N} \wedge t_j \in t_i + \Delta$.

(a) *Two interactive objects.* We analyse $[\![Score_{I,S,\mathrm{Pr},R}]\!]$, which is the intersection of the denotations of $iEvent_{i,\mathrm{Pr}}$, $iEvent_{j,\mathrm{Pr}}$, $Clock(0)$ and $!\mathbf{tell}(0 < p_i \leq n_\infty)$. Since $iEvent_{i,\mathrm{Pr}}$ has no predecessors, the constraint $\bigwedge_{j\in \mathrm{Pr}(i)}\ launched_j$ can always be deduced. Whenever ev_i can be deduced (after the first time unit), $launch_i$ can also be deduced. Similarly, whenever ev_j can be deduced, $launch_j$ can also be deduced, and this can only happen after $launched_i$ can be deduced. The constraint $clock + 1 > p_i$ can be deduced for values of p_i that satisfy the constraints $0 < p_i < n_\infty$ and $p_j = p_i + \Delta$. Finally, $clock + 1 > p_j$ can be deduced when $clock = n_\infty - 1$ can be deduced because $0 < p_j < n_\infty$. Therefore, the values of T_i and T_j are given by $T_i \subseteq [1, n_\infty] \wedge T_j \subseteq [1, n_\infty] \wedge T_j \in T_i + \Delta$.

(b) *Two static events.* We analyse $[\![Score_{I,S,\mathrm{Pr},R}]\!]$, which is the intersection of the denotations of $sEvent_{i,\mathrm{Pr}}$, $sEvent_{j,\mathrm{Pr}}$, $Clock(0)$ and $!\mathbf{tell}(0 < p_i \leq n_\infty)$. Figure A3 presents the denotation of $sEvent_{i,\mathrm{Pr}}$.

(i) Since e_i has no predecessors, the constraint $\bigwedge_{j\in \mathrm{Pr}(i)}\ launched_j$ can always be deduced. It is trivial to show that the constraint $clock + 1 < p_i$; therefore, $launch_i$ will be deduced in the second time unit.

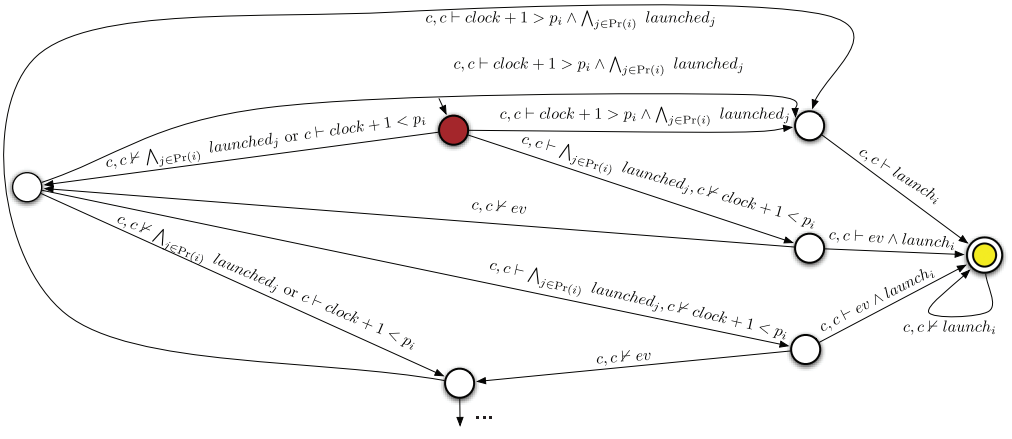

Figure A2. Denotation of process $iEvent_{i,\mathrm{Pr}}$ represented as a Büchi automaton, $c \in S \subset_{fin} C$ (i.e. a finite subset of C).

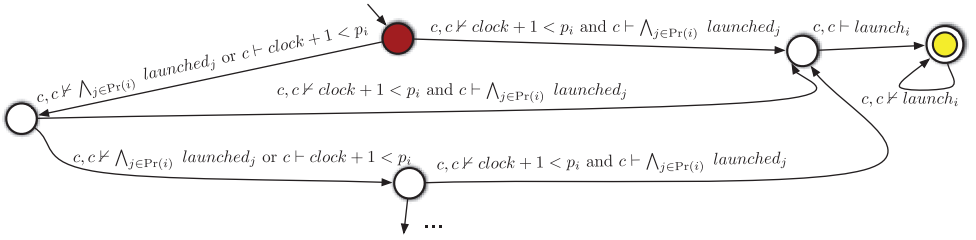

Figure A3. Denotation of process $sEvent_{i,\Pr}$ represented as a Büchi automaton, $c \in S \subset_{fin} C$.

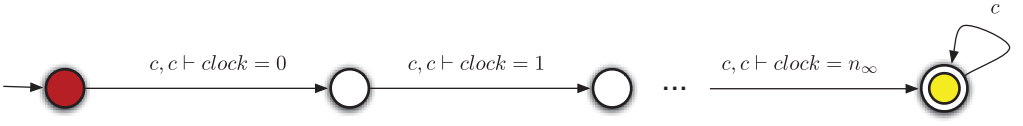

Figure A4. Denotation of process $Clock(0)$ represented as a Büchi automaton, $c \in S \subset_{fin} C$.

Table A5. Temporal constraints to launch the events in the `ntcc` model Vs. Temporal constraints of the event structures semantics. We denote an interactive object as $((i))$ and a static one as (i).

ε^*	Values of T_i for process $Score_{\varepsilon^*}$	$\mathrm{tc}(\varepsilon^*)$
$((i))$	$T_i \subseteq [1, n_\infty]$	$t_i \in \mathbb{N}$
$(i) \overset{\Delta}{\mapsto} (j)$	$T_i = \{1\} \wedge T_j \in T_i + \min(\Delta)$	$t_i \in \mathbb{N} \wedge t_j \in t_i + \Delta$
$((i)) \overset{\Delta}{\mapsto} (j)$	$T_i \subseteq [1, n_\infty] \wedge T_j \in T_i + \min(\Delta)$	$t_i \in \mathbb{N} \wedge t_j \in t_i + \Delta$
$((i)) \overset{\Delta}{\mapsto} ((j))$	$T_i \subseteq [1, n_\infty] \wedge T_j \in T_i + \Delta$	$t_i \in \mathbb{N} \wedge t_j \in t_i + \Delta$
$(i) \overset{\Delta}{\mapsto} ((j))$	$T_i = \{1\} \wedge T_j \in T_i + \Delta$	$t_i \in \mathbb{N} \wedge t_j \in t_i + \Delta$

(ii) Once the constraint $launch_i$ can be deduced in the second time unit, constraints $launched_i$ and $p_i = 1$ will be deduced from all subsequent sequences, thus the constraint $clock + 1 < p_j$ will be deduced from the store until $clock = 1 + \min(\Delta)$; therefore, the constraint $launch_j$ will be deduced in the time unit $1 + \min(\Delta)$.

Therefore $T_i = \{1\}$ and $T_j = \{1 + \min(\Delta)\}$.

(c) *A static event followed by an interactive object.* We analyse $[\![Score_{I,S,\Pr,R}]\!]$, which is the intersection of the denotations of $sEvent_{i,\Pr}$, $iEvent_{j,\Pr}$, $Clock(0)$ and $!\mathbf{tell}(0 < p_i \leq n_\infty)$. The constraint $launch_i$ can be deduced in position one for process $sEvent_{i,\Pr}$, as it was described in part (2)(b)i. The positions where $launch_j$ can be deduced are in the set $T_j = [1 + \min(\Delta), n_\infty)$. The reason is that the constraint $clock + 1 < p_j$ cannot be deduced before $clock = 1 + \min(\Delta)$ and the constraint $clock + 1 > p_j$ can only be deduced when $clock = n_\infty$.

(d) *An interactive object followed by a static event.* We analyse $[\![Score_{I,S,\Pr,R}]\!]$, which is the intersection of the denotations of $iEvent_{i,\Pr}$, $sEvent_{j,\Pr}$, $Clock(0)$ and $!\mathbf{tell}(0 < p_i \leq n_\infty)$. The positions where $launch_i$ can be deduced are the ones described in part (1). Once $launched_i$ can be deduced from a store, the constraint $launch_j$ will be deduced from all stores $\min(\Delta)$ time units after, because $clock + 1 < p_i$ cannot be deduced anymore; thus, $T_i = [1, n_\infty), T_j \in T_i + \min(\Delta)$.

(3) *Inductive case.* We must prove that if the proposition holds for an event structure ε^* with k events, it also holds for ε'^*, which is ε^* with one more event. Let e_{k+1} be the new event in ε'^*, $\mathrm{pred}(e_{k+1})$ the event delays between e_{k+1} and its predecessors and $\mathrm{succ}(e_{k+1})$ the event delays between e_{k+1} and its successors. We know by inductive hypothesis that the preposition holds for a ε^* with k events, we must prove that adding the event e_{k+1} to the events of ε^* and $\mathrm{pred}(e_{k+1})$ and $\mathrm{succ}(e_{k+1})$ to the event delays of ε^*, the proposition also holds. If the new event has no predecessors nor successors, the proof continues as part (1). Otherwise, for each predecessor and each successor, the proof continues as part (2) because the event structure is *dispatchable*. Therefore, all tuples in $T_1 \times T_2 \cdots \times T_{k+1}$ are solutions of $\mathrm{tc}(\varepsilon'^*)$, which is a contradiction.

■