Formal Methods In System Design, 12, 189–216 (1998)

A Consistent Causality-Based View on a Timed Process Algebra Including Urgent Interactions

JOOST-PIETER KATOEN katoen@informatik.uni-erlangen.de

Universität Erlangen-Nürnberg, Institut für Informatik VII, Martensstrasse 3, D-91058 Erlangen, Germany

ROM LANGERAK langerak@cs.utwente.nl

ED BRINKSMA brinksma@cs.utwente.nl

Faculty of Computing Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands

DIEGO LATELLA d.latella@cnuce.cnr.it

CNR, Ist. CNUCE, Via Santa Maria 36, 56100 Pisa, Italy

TOMMASO BOLOGNESI bolognesi@iei.pi.cnr.it

CNR, Ist. Elaborazione dell'Informazione, Via Santa Maria 46, 56100 Pisa, Italy

Abstract. This paper discusses a timed variant of a process algebra akin to LOTOS, baptized UPA, in a causality-based setting. Two timed features are incorporated—a delay function which constrains the occurrence time of atomic actions and an urgency operator that forces (local or synchronized) actions to happen urgently. Timeouts are typical urgent phenomena. A novel timed extension of event structures is introduced and used as a vehicle to provide a denotational causality-based semantics for UPA. Recursion is dealt with by using standard fixpoint theory. In addition, an operational semantics is presented based on separate time- and action-transitions that is shown to be consistent with the event structure semantics. An interleaving semantics for UPA is immediately obtained from the operational semantics. By adopting this dual approach the well-developed timed interleaving view is extended with a consistent timed partial order view and a comparison is facilitated of the partial order model and the variety of existing (interleaved) timed process algebras.

Keywords: causality, consistency of semantics, event structure, LOTOS, process algebra, semantics, time, true concurrency, urgency

1. Introduction

We study—in a causality-based setting—a timed extension of a basic process algebraic formalism including multi-way synchronisation. The formalism, referred to as UPA, is based on the core of LOTOS [5], i.e., LOTOS without data types and value passing. The approach followed in this paper can, however, be adapted to related process algebras like CCS [27], CSP [17], and ACP [3]. Two timed features are incorporated—a delay function which constrains the occurrence time of atomic actions and an urgency operator that forces (local or synchronized) actions to happen urgently. Urgent actions are important to model timeouts that are forced to occur at a certain time—irrespective of the rest of the system—in case some desired action (like receiving an acknowledgement) has not happened yet.

Various timed process algebras have been developed based on the *interleaving* of independent actions [6, 28, 35]. Although each timed formalism has its own characteristics and operators, one may say that the way in which to construct a timed process algebra in an interleaving setting is well-developed, see for instance the recipe in [30]. Due to their observational nature interleaving models are quite appropriate for the description of a system at a high level of abstraction (i.e., considering the system's behaviour as viewed from the outside), and for conformance testing [1]. The incorporation of time in such models is important to obtain an overall view on how the system's behaviour evolves in (linear) time. In the final stages of the design trajectory, however, the global state assumption hampers us to faithfully model the distribution aspects of a system, each part having its own local state. At this design phase the 'local' causal dependencies between actions and their timing constraints are important, while interleavings with actions of other (irrelevant) system parts burden the design. In particular, if the specification serves as a prescription for the system's implementation rather than as a description of the observational behaviour of a system, interleaving models become unattractive or even misleading since the independence of actions is not reflected properly, see [36]. (Timed) partial-order models are considered to be much more appropriate here.

This motivates the need for the support of the design process with a coherent set of complementary semantic models. For our timed formalism UPA we therefore take a dual approach—we provide an event-based operational semantics for this timed process algebra which yields an interleaving semantics when omitting the event identifiers, and extend this view with a novel *causality-based* semantics. The resulting operational and denotational semantics are proven to be consistent in the sense that they generate identical sets of timed event traces. The causality-based model is a timed extension of Langerak's *extended bundle* event structures [23], an adaptation of Winskel's labelled event structures [38] to fit the specific requirements of parallel composition with multi-way synchronisation and the requirements of disruption ($[>$).

The specification of timing aspects is crucial for performing performance analysis. Preliminary studies indicate that the analysis of performance aspects could benefit from a causality-based setting [10, 11, 21] since the parallelism between system components is explicitly retained in the semantic model. In addition, a causality-based model facilitates the possibility to study only that part of a system in which one is interested for the analysis in a relatively easy way (locality) and does not suffer from the *state explosion problem*—parallelism leads to the sum of the components states, rather than to their product (as in interleaving).

2. A temporal process algebra

This paper is based on the process algebraic language PA, in fact LOTOS with a somewhat more concise syntax, generated by the following grammar:

$$B ::= \mathbf{0} \mid \surd \mid a\,;\, B \mid B + B \mid B \,||_G\, B \mid B[H] \mid B \setminus G \mid B >> B \mid B \,[>\, B \mid P.$$

Table 1. Structured operational semantics of PA.

	$\vdash$	$\surd \xrightarrow{\delta} \mathbf{0}$
	$\vdash$	$a\,;\, B \xrightarrow{a} B$
$B_1 \xrightarrow{a} B_1'$	$\vdash$	$B_1 + B_2 \xrightarrow{a} B_1'$
$B_2 \xrightarrow{a} B_2'$	$\vdash$	$B_1 + B_2 \xrightarrow{a} B_2'$
$B_1 \xrightarrow{a} B_1' \quad a \neq \delta$	$\vdash$	$B_1 >> B_2 \xrightarrow{a} B_1' >> B_2$
$B_1 \xrightarrow{\delta} B_1'$	$\vdash$	$B_1 >> B_2 \xrightarrow{\tau} B_2$
$B_1 \xrightarrow{a} B_1' \quad a \neq \delta$	$\vdash$	$B_1 [> B_2 \xrightarrow{a} B_1' [> B_2$
$B_1 \xrightarrow{\delta} B_1'$	$\vdash$	$B_1 [> B_2 \xrightarrow{\delta} B_1'$
$B_2 \xrightarrow{a} B_2'$	$\vdash$	$B_1 [> B_2 \xrightarrow{a} B_2'$
$B_1 \xrightarrow{a} B_1' \quad a \notin G^\delta$	$\vdash$	$B_1 \,\|\|_G\, B_2 \xrightarrow{a} B_1' \,\|\|_G\, B_2$
$B_2 \xrightarrow{a} B_2' \quad a \notin G^\delta$	$\vdash$	$B_1 \,\|\|_G\, B_2 \xrightarrow{a} B_1 \,\|\|_G\, B_2'$
$B_1 \xrightarrow{a} B_1' \ \wedge\ B_2 \xrightarrow{a} B_2' \quad a \in G^\delta$	$\vdash$	$B_1 \,\|\|_G\, B_2 \xrightarrow{a} B_1' \,\|\|_G\, B_2'$
$B \xrightarrow{a} B' \quad a \notin G$	$\vdash$	$B \setminus G \xrightarrow{a} B' \setminus G$
$B \xrightarrow{a} B' \quad a \in G$	$\vdash$	$B \setminus G \xrightarrow{\tau} B' \setminus G$
$B \xrightarrow{a} B'$	$\vdash$	$B[H] \xrightarrow{H(a)} B'[H]$
$B \xrightarrow{a} B' \quad P := B$	$\vdash$	$P \xrightarrow{a} B'$

We assume a given set of observable actions Act and an additional *invisible action* τ; $\tau \notin$ Act. $\mathbf{0}$ denotes *inaction*; $\surd$ represents the *successful termination* process. $a\,;\, B$ denotes the *action-prefix* of $a \in \mathsf{Act} \cup \{\,\tau\,\}$ and B. The *choice* between B_1 and B_2 is denoted $B_1 + B_2$ and their *sequential composition* by $B_1 >> B_2$. $B_1 \,||_G\, B_2$ denotes *parallel composition* where actions in G ($G \subseteq$ Act) are synchronisation actions. $|||$ abbreviates $||_\varnothing$, i.e., parallel composition without synchronisation. $B[H]$ denotes the *relabelling* of B according to H where $H : \mathsf{Act} \longrightarrow \mathsf{Act}$. $B \setminus G$ denotes *hiding*, with $G \subseteq$ Act. $B_1 [> B_2$ denotes the *disruption* of B_1 by B_2; i.e., B_1 may at any point of its execution disrupted by B_2, unless it successfully terminated. Finally, P denotes a *process instantiation* where a behaviour is considered in the context of a set of process definitions of the form $P := B$ where B possibly contains occurrences of P.

The precedences of the composition operators are, in decreasing binding order: $;$, $+$, $||$, $[>$, $>>$, $\setminus$ and $[\,]$. Trailing **0**s are usually omitted.

The standard (interleaving) semantics of PA is presented in Table 1, in the style of [33]. The special action δ indicates the *successful termination* action of a behaviour; we assume $\delta \notin$ Act. All relabelling functions H are extended to $\mathsf{Act} \cup \{\,\tau, \delta\,\}$ under the requirement that $H(\tau) = \tau, H(\delta) = \delta$ and for $a \in$ Act we have $H(a) \notin \{\,\tau, \delta\,\}$. G^δ denotes $G \cup \{\,\delta\,\}$. Note that in $B_1 \,||_G\, B_2$ the component behaviours always synchronize on successful termination action δ.

The temporal variant of PA, baptized UPA, is generated by the grammar:

$$B ::= \mathbf{0} \;\big|\; \surd \;\big|\; (t)\, a\,;\, B \;\big|\; B + B \;\big|\; B \,||_G\, B \;\big|\; B[H] \;\big|\; B \setminus G \;\big|\; B >> B \;\big|\; B [> B \;\big|\; \mathcal{U}_U(B) \;\big|\; P.$$

We use $\mathsf{Time} = \mathbb{R}^+ \cup \{\,0, \infty\,\}$ as time domain and t to range over Time. $(t)\, a\,;\, B$ denotes the *timed action-prefix* of a and B where a is allowed (but not forced) to occur at some

$t' \geqslant t$. We write a for $(0)\ a$. Actions are atomic and occur instantaneously. $\mathcal{U}_U(B)$ behaves like B except that actions in U, $U \subseteq \mathsf{Act} \cup \{\tau\}$, are *forced* to happen as soon as they are enabled. Actions in U different from τ are visible to the environment but the environment cannot synchronize with them. If G, U are singleton sets, $\{a\}$ say, we simply write $||_a$, $\backslash a$ and $\mathcal{U}_a()$. t_a denotes the time of occurrence of a.

Behaviours may synchronize on a common action as soon as all participants are ready to engage in it, i.e., when all individual timing constraints on such action are met. E.g., action c is enabled in $a\,;\ (3)\ c\,||_c\,b\,;\ (7)\ c$ if both a has occurred at least 3 time units before and b has occurred at least 7 time units before, i.e., $t_c \geqslant max(t_a{+}3, t_b{+}7)$. In $a\,;\ (t_1)\ b\,||_{\{a,b\}}\,a\,;\ (t_2)\ b$ action b is enabled after $t_a{+}max(t_1, t_2)$.

The notion of urgency here is an extension of the notion of urgency in an earlier paper [10] where urgent actions are assumed to model activities whose occurrence can be controlled completely internally. Here, urgency can involve several participants and is strongly influenced by the notion of urgency in [6, 7] (see also later on). Once made urgent, actions cannot be used for synchronisation any further. Without such a restriction, expressions like $B = \mathcal{U}_b((2)\ b)\,||_b\,\mathcal{U}_b((1)\ b)$ would be allowed. Conforming to the principle that an urgent action happens as soon as all participants are ready for it, $(b, 2)$ would be a trace of B. This would cause a delay of action b in the right component, contradicting its local urgency. The fact that we do not allow synchronisations on urgent events is captured by a syntactical constraint on behaviours which can easily be formulated [20] and is omitted here. (An alternative view is to only allow processes like $\mathcal{U}_a(B)$ to be embedded in a context that is always willing to perform a actions, i.e., without imposing additional timing constraints. The observability of a could then be exploited.)

Urgent actions are forced to happen as soon as all participants are ready for it. E.g., in $B = a\,;\ (3)\ c\,||_c\,b\,;\ ((2)\ d + (5)\ c)$ action c can occur at any $t_c \geqslant max(t_a{+}3, t_b{+}5)$ if d has not yet appeared. If c has not yet occurred, d can occur from $t_b{+}2$ on. In $\mathcal{U}_c(B)$ action c is forced to happen at $t_c = max(t_a{+}3, t_b{+}5)$ in case d has not yet appeared at that time. That is, d is prevented to occur at any time later than t_c, and can only occur in the interval $[t_b{+}2, t_c]$. At time t_c a non-deterministic choice between c and d occurs (so-called weak timeout [32])—urgency does not impose a priority in this case.

The urge operator is inspired by a similar operator, denoted ρ, introduced in [6]. ρ prevents the passage of time as an alternative to the occurrence of an enabled urgent action. [6] allows synchronisations on urged actions. Such synchronisations only succeed if all participants are ready to participate at the same instant of time. In case a synchronisation does not succeed, a *time deadlock* appears, a situation in which passage of time is blocked as a result of which the entire system halts execution. In our semantic models no notion of time deadlock is possible. [7] generalises the notion of urgency by introducing the **time** operator. **time** $a(t_1, t_2)$ **in** B denotes behaviour B in which a must occur in interval $[t_1, t_2]$ once it is enabled. $\mathcal{U}_a(B)$ is akin to (**time** $a(0, 0)$ **in** B) $\backslash a$, the main difference is that the former performs visible a actions, while the latter turns a actions into internal actions.

3. Extended bundle event structures

Extended bundle event structures (or, simply: event structures) [23] consist of *events* labelled with actions (an event modelling the occurrence of its action), together with relations of causality and conflict between events. System runs can be modelled as partial orders of events satisfying certain constraints posed by the causality and conflict relations between the events.

An *asymmetric conflict* is a binary relation, denoted $\rightsquigarrow$, between events and the intended meaning of $e \rightsquigarrow e'$ is that (i) if e' occurs it disables the occurrence of e, and (ii) if e and e' both occur in a single system run then e causally precedes e'. Notice that it is *not* required for $\rightsquigarrow$ to be symmetric, hence the name 'asymmetric', which, in this context, does not mean that $e \rightsquigarrow e' \;\Rightarrow\; e' \not\rightsquigarrow e$ as it might suggest. $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$ is allowed and is equivalent with $e \# e'$, the usual symmetric conflict in event structures[1].

Causality is represented by a binary relation, the *bundle relation*, denoted by $\mapsto$. Given a set X of events, that are pairwise in conflict, and an event e, the interpretation of $X \mapsto e$ is that if e happens in a system run, exactly one event in X has happened before (and caused e). This enables us to uniquely define a causal ordering between the events in a system run. Set X is called the *bundle set*. When there is neither a conflict nor a causal relation between events they are independent. Once enabled, independent events can occur in any order or in parallel.

Definition 1. An *(extended bundle) event structure* $\mathcal{E}$ is a quadruple $(E, \rightsquigarrow, \mapsto, l)$ with E, a set of *events*, $\rightsquigarrow \subseteq E \times E$, the (irreflexive) *asymmetric conflict* relation, $\mapsto \subseteq \mathcal{P}(E) \times E$, the *bundle* relation, and $l : E \longrightarrow L$, the *action-labelling* function, where L is a set of action labels, such that

$$\forall\, X \subseteq E, e \in E : X \mapsto e \;\Rightarrow\; (\forall\, e', e'' \in X : e' \neq e'' \;\Rightarrow\; e' \rightsquigarrow e'').$$

The constraint specifies that for bundle $X \mapsto e$ all events in X are in mutual conflict. Event structures are graphically represented in the following way. Events are denoted as dots; near the dot the action label is given. $e \rightsquigarrow e'$ is indicated by a dotted arrow from e to e'; if also $e' \rightsquigarrow e$, then a dotted line is drawn instead. A bundle $X \mapsto e$ is indicated by drawing an arrow from each event in X to e and connecting all arrows by small lines. We denote an event e labelled a by e_a. EBES denotes the class of event structures; $\mathcal{E}$ ranges over EBES.

In the sequel we adopt the following notations. For sequences $\sigma = x_1 \ldots x_n$, let $\overline{\sigma}$ denote the set of elements in σ, i.e., $\overline{\sigma} \triangleq \{ x_1, \ldots, x_n \}$. ε denotes the empty sequence. For non-empty sequence σ, let σ_i denote the prefix of σ up to the $(i{-}1)$-th element, i.e., $\sigma_i \triangleq x_1 \ldots x_{i-1}$, for $0 < i \leqslant n{+}1$. For σ a sequence of events $e_1 \ldots e_n$ we define $\mathsf{cfl}(\sigma) \triangleq \{ e \in E \mid \exists\, e_i \in \overline{\sigma} : e \rightsquigarrow e_i \}$ and $\mathsf{sat}(\sigma) \triangleq \{ e \in E \mid \forall X \subseteq E : X \mapsto e \;\Rightarrow\; X \cap \overline{\sigma} \neq \varnothing \}$. $\mathsf{cfl}(\sigma)$ is the set of events that are disabled by some event in σ. $\mathsf{sat}(\sigma)$ is the set of events that have a causal predecessor in σ for all bundles pointing to them. That

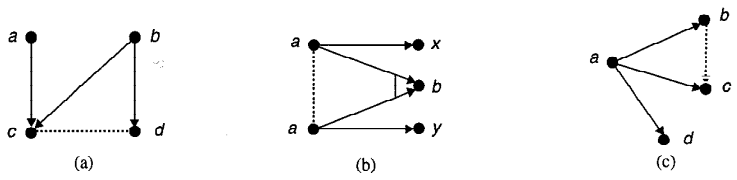


Figure 1. Some example event structures.

is, for events in $\mathsf{sat}(\sigma)$ all bundles are 'satisfied'. The set of events 'enabled' by σ, $\mathsf{en}(\sigma)$, is defined as $\mathsf{en}(\sigma) \triangleq \mathsf{sat}(\sigma) \setminus (\mathsf{cfl}(\sigma) \cup \overline{\sigma})$.

Event traces consist of distinct events ($e_i \notin \overline{\sigma_i}$) and are conflict-free ($e_i \notin \mathsf{cfl}(\sigma_i)$), for obvious reasons. In addition, each event in the event trace is preceded in the sequence by a causal predecessor for each bundle pointing to it ($e_i \in \mathsf{sat}(\sigma_i)$).

Definition 2. An *event trace* σ of $\mathcal{E}$ is a sequence of events $e_1 \ldots e_n$ with $e_i \in \mathsf{en}(\sigma_i)$, for all $0 < i \leqslant n$. Let $T(\mathcal{E})$ denote the set of event traces of $\mathcal{E}$.

EXAMPLE: Figure 1(a) has bundles $\{ e_a \} \mapsto e_c$, $\{ e_b \} \mapsto e_c$, $\{ e_b \} \mapsto e_d$, and a symmetric conflict between e_c and e_d. Some event traces of Figure 1(a) are $e_a\, e_b\, e_c$, $e_b\, e_d\, e_a$ and $e_b\, e_a$. In Figure 1(b) we have $\{ e_a, e'_a \} \mapsto e_b$, $\{ e_a \} \mapsto e_x$ and $\{ e'_a \} \mapsto e_y$. Figure 1(c) has asymmetric conflict $e_b \rightsquigarrow e_c$. $e_a\, e_b\, e_c$ and $e_a\, e_c$ are event traces of this event structure. □

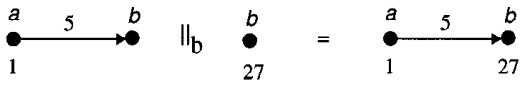
Event structures can be used to provide a noninterleaving semantics to PA in a compositional way. For finite behaviours this is defined in the appendix. The expressions corresponding to Figure 1 are as follows:

- (a) $a\,;\, c \,||_{\{\, c \,\}}\, b\,;\, (c + d)$,
- (b) $(a\,;\, x \,|||\, a\,;\, y) \,||_{\{\, a \,\}}\, (a\,;\, b)$, and
- (c) $a\,;\, ((b \,[\!>\, c) \,|||\, d)$.

4. Urgent event structures

Time is added to bundle event structures in two ways. To specify the relative delay between causally dependent events time is associated to bundles, and in order to facilitate the specification of timing constraints on events that have no bundle pointing to them (i.e., the initial events), time is also associated to events.[2] Though it seems sufficient to only have time labels for initial events, synchronisation of events makes it necessary to allow for equipping all events with time labels, including the non-initial ones.

Consider, e.g.,



where the result says that if e_a occurs at t_a then e_b is enabled from $max(t_a{+}5, 27)$.

We assume mappings $\mathcal{T}$ and $\mathcal{D}$ to associate a value of Time, the time domain, to bundles and events, respectively. A bundle $X \mapsto e$ with $\mathcal{T}((X, e)) = t$ is denoted by $X \stackrel{t}{\mapsto} e$; its interpretation is that if an event in X has happened at a certain time, then e is enabled t time units later. For events that have more than one bundle pointing to them we take the following interpretation. Consider $\{ e_a \} \stackrel{t}{\mapsto} e_c$ and $\{ e_b \} \stackrel{t'}{\mapsto} e_c$. Then, if e_a happens at time t_a and e_b at t_b, then e_c is enabled at $max(t_a{+}t, t_b{+}t')$. $\mathcal{D}$ associates time to events; $\mathcal{D}(e) = t$ denotes that e can happen after t time units from the beginning of the system, usually assumed to be time 0.

Bundle and event delays determine the minimal time at which an event can occur. In order to force events to occur once they are enabled we use *urgent events*. Urgency is modelled by a predicate $\mathcal{U}$: $\mathcal{U}(e)$ is true iff e is an urgent event.

Definition 3. A *urgent* event structure Ψ is a quadruple $\langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ with $\mathcal{E}$ an event structure $(E, \rightsquigarrow, \mapsto, l)$, $\mathcal{T} : \mapsto \longrightarrow$ Time, the *timing* function, $\mathcal{D} : E \longrightarrow$ Time, the *delay* function, and $\mathcal{U} : E \longrightarrow$ Bool, the *urgency* predicate.

For depicting urgent event structures we use the following conventions. The time associated with a bundle and event is a non-negative real and is depicted near to a bundle and event, respectively. For convenience, zero delays are omitted. Urgent events are depicted by open dots, and ordinary events by closed dots. UES denotes the class of urgent event structures; Ψ ranges over UES.

EXAMPLE: Some example urgent event structures are depicted in Figure 2. Figure 2(a) has bundles $\{ e_a \} \stackrel{3}{\mapsto} e_c$, $\{ e_b \} \stackrel{5}{\mapsto} e_c$, $\{ e_b \} \stackrel{2}{\mapsto} e_d$, and a conflict between urgent event e_c and

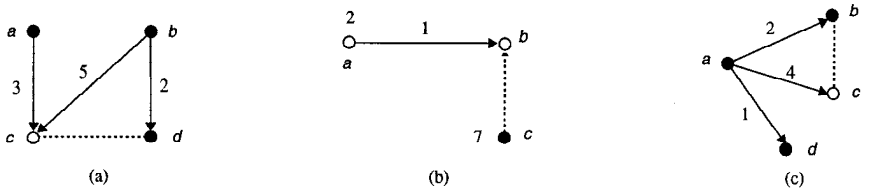


Figure 2. Some example urgent event structures.

e_d. For Figure 2(b) we have $\mathcal{D}(e_a) = 2$, $\mathcal{D}(e_b) = 0$, $\mathcal{D}(e_c) = 7$ and $\mathcal{T}((\{\, e_a \,\}, e_b)) = 1$. □

As a generalisation of the notion of event trace we define the notion of *timed event trace*. A timed event (e, t) denotes that e happened at time t. For sequences of timed events $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ let $[\sigma]$ denote the sequence of events in σ, i.e., $[\sigma] \triangleq e_1 \ldots e_n$. Let $\mathsf{time}(\sigma, e)$ denote the time instant from which $e \in \mathsf{en}([\sigma])$ can happen, given that each event e_i in σ occurred at time t_i. Event e can occur if (i) its absolute delay $\mathcal{D}(e)$ is respected, (ii) the time relative to all its immediate causal predecessors is respected, and (iii) for each event e_j with $e_j \rightsquigarrow e$ we have that e occurs at at least t_j. (ii) and (iii) take care of the fact that events cannot occur before their causes, entailing that causal ordering implies temporal ordering. So, $\mathsf{time}(\sigma, e)$ is obtained by taking the maximum of $\mathcal{D}(e)$ with proper sets (the elements of H_1 and H_2 below) representing the constraints (ii) and (iii):

$$\begin{aligned} \mathsf{time}(\sigma, e) \triangleq\ & Max(\{\, \mathcal{D}(e) \,\} \cup H_1 \cup H_2) \text{ where} \\ & H_1 = \{\, t_j + t \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \ \wedge\ X \cap \overline{[\sigma]} = \{\, e_j \,\} \,\} \\ & H_2 = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow e \,\} \quad . \end{aligned}$$

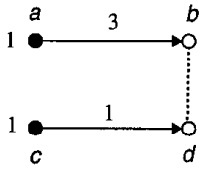
Definition 4. A *timed event trace* of $\langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ is a sequence σ of timed events $(e_1, t_1) \ldots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$, satisfying $e_1 \ldots e_n \in T(\mathcal{E})$, $i < j \ \Rightarrow\ t_i \leqslant t_j$, for all $0 < i, j \leqslant n$, and

1. $\forall\, i : (\, \neg \mathcal{U}(e_i) \ \Rightarrow\ t_i \geqslant \mathsf{time}(\sigma_i, e_i)) \ \wedge\ (\mathcal{U}(e_i) \ \Rightarrow\ t_i = \mathsf{time}(\sigma_i, e_i))$, and
2. $\forall\, i, e : (e \in \mathsf{en}(\sigma_i) \ \wedge\ \mathcal{U}(e)) \ \Rightarrow\ t_i \leqslant \mathsf{time}(\sigma_i, e)$.

For $\Psi \in \mathsf{UES}$ we denote the set of timed event traces of Ψ by $T_U(\Psi)$.

The first constraint requires correct times to be associated to events in σ—ordinary events can happen at any moment from the time they are enabled and urgent events can happen only as soon as they are enabled. This constraint does, however, not take into account the fact that urgent events may prevent other events to occur after a certain time. E.g., according to the first constraints, Figure 2(a) has timed event trace $(e_a, 0)\,(e_b, 2)\,(e_d, 8)$. However, if e_d has not happened before time $max(0, 0{+}3, 2{+}5) = 7$, then urgent event e_c should have happened at time 7. Thus, $(e_a, 0)\,(e_b, 2)\,(e_d, 8)$ should *not* be considered a legal timed event trace. The last constraint takes this matter into account.

We like to point out that time-consistency, i.e., $i < j \Rightarrow t_i \leqslant t_j$, is essential in a context in which in principle any event can be declared to be urgent. The reason for this is that urgency is an intrinsically *global* property: the fact that some event e is urgent influences for events, which seem at first sight completely independent from e, the ability to appear at a certain time instant. So, in order to decide whether an event may happen it is necessary to know in the entire system which events have happened already (in time). E.g., if we would omit the time-consistency requirement, then



would have trace $(e_a, 1)\,(e_b, 4)\,(e_c, 2)$, whereas if e_c happens at time 2 urgent event e_d is forced at time 3 and should disable the occurrence of e_b. In [22] we showed that a somewhat restricted, but still very useful, form of urgency reduces the global impact of urgent events and can avoid this time-consistency requirement.

EXAMPLE: For the following sequences of timed events the conditions are given under which they are timed event traces of Figure 2(a):

$$(e_a, t_a)\,(e_b, t_b)\,(e_d, t_d) \;\text{ if }\; t_a \leqslant t_b \;\wedge\; t_b{+}2 \leqslant t_d \leqslant max(t_a{+}3, t_b{+}5), \text{ and}$$
$$(e_a, t_a)\,(e_b, t_b)\,(e_c, t_c) \;\text{ if }\; t_a \leqslant t_b \;\wedge\; t_c = max(t_a{+}3, t_b{+}5).$$

The only maximal timed event trace of Figure 2(b) is $(e_a, 2)\,(e_b, 3)$. Here, e_c can never happen, since after the occurrence of e_a (which will be forced at time 2) e_b will occur (at time 3), so excluding e_c. Thus, e_a excludes e_c though they seem to be completely independent! It appears that the asymmetric conflict between e_c and e_b 'propagates back' to an asymmetric conflict between e_a and e_c.

Figure 2(c) models a typical timeout scenario: after the occurrence of e_a at time t_a event e_b is enabled from $t_a{+}2$, but if it does not occur at $t_a{+}4$, then e_c is forced to occur (at $t_a{+}4$). □

5. Causality-based semantics of UPA

This section presents a causality-based semantics for UPA using urgent event structures. We define a mapping $\mathcal{E}_U[\![\ ]\!] : \mathsf{UPA} \longrightarrow \mathsf{UES}$. For convenience we use the denotational semantics $\mathcal{E}'[\![\ ]\!]$ for the untimed case which is defined in the appendix. Recursion is dealt with in Section 6.

Definition 5. $\Phi : \mathsf{UPA} \longrightarrow \mathsf{PA}$ is defined as follows:

$$\begin{aligned}
\Phi(\mathbf{0}) &\triangleq \mathbf{0}\\
\Phi(\surd) &\triangleq \surd\\
\Phi((t)\,a\,;\,B) &\triangleq a\,;\,\Phi(B)\\
\Phi(B_1\ \mathsf{op}\ B_2) &\triangleq \Phi(B_1)\ \mathsf{op}\ \Phi(B_2) \text{ for } \mathsf{op} \in \{+, ||_G, >>, [>\}\\
\Phi(\mathsf{op}\ B) &\triangleq \mathsf{op}\ \Phi(B) \text{ for } \mathsf{op} \in \{\backslash, [\,]\}\\
\Phi(\mathcal{U}_U(B)) &\triangleq \Phi(B).
\end{aligned}$$

$\Phi(B)$ is the untimed behaviour corresponding to B obtained by omitting all time and urgency annotations in B.

In the rest of this section let $\mathcal{E}_U[\![B_i]\!] = \Psi_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$, for $i = 1, 2$, with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ and $E_1 \cap E_2 = \varnothing$. The functions *init* and *exit* which denote the set of initial and termination events, respectively, are defined for event structures in the appendix and are used for urgent event structures in the same way, that is, $init(\Psi_i) \triangleq init(\mathcal{E}_i)$ and $exit(\Psi_i) \triangleq exit(\mathcal{E}_i)$. Let E_U denote the (infinite) universe of events and $pos(\Psi)$ denote the set of non-zero delay events in Ψ, i.e., $pos(\Psi) \triangleq \{\, e \in E \mid \mathcal{D}(e) \neq 0 \,\}$. We abbreviate $pin(\Psi) = pos(\Psi) \cup init(\Psi)$.

Definition 6. $\mathcal{E}_U[\![\ ]\!] : \mathsf{UPA} \longrightarrow \mathsf{UES}$ is defined for **0**, $\surd$, and $(t)\ a\,;$ as follows:

$$
\begin{array}{rcl}
\mathcal{E}_U[\![\mathbf{0}]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi(\mathbf{0})]\!], \varnothing, \varnothing, \varnothing \rangle \\
\mathcal{E}_U[\![\surd]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi(\surd)]\!], \{\, (e_\delta, 0) \,\}, \varnothing, \{\, (e_\delta, \text{false}) \,\} \rangle \\
\mathcal{E}_U[\![(t)\ a\,;\ B_1]\!] & \triangleq & \langle (E, \rightsquigarrow_1, \mapsto, l_1 \cup \{\, (e_a, a) \,\}), \mathcal{D}, \mathcal{T}, \mathcal{U}_1 \cup \{\, (e_a, \text{false}) \,\} \rangle \text{ with} \\
E & = & E_1 \cup \{\, e_a \,\} \text{ for } e_a \in E_U \setminus E_1 \\
\mapsto & = & \mapsto_1 \ \cup\ (\{\, \{\, e_a \,\} \,\} \times pin(\Psi_1)) \\
\mathcal{D} & = & \{\, (e_a, t) \,\} \cup (E_1 \times \{\, 0 \,\}) \\
\mathcal{T} & = & \mathcal{T}_1 \cup \{\, ((\{\, e_a \,\}, e), \mathcal{D}_1(e)) \mid e \in pin(\Psi_1) \,\}.
\end{array}
$$

The semantics of **0** and $\surd$ is self-explanatory. For $(t)\ a\,;\ B_1$ a bundle is introduced from a new non-urgent event e_a (labelled a) to all initial events of Ψ_1 (as e_a causally precedes them) and all events in Ψ_1 that have a non-zero delay. For all these initial and non-zero delay events e the delay is now relative to e_a, so each bundle $\{\, e_a \,\} \mapsto e$ is associated with a time delay $\mathcal{D}_1(e)$, and $\mathcal{D}(e)$ is made zero. $\mathcal{D}(e_a)$ becomes t. In the untimed case it suffices to only introduce bundles from e to the initial events of Ψ_1, cf. the appendix. The bundles to all positive events of Ψ_1 that are introduced in the timed case are used for the sole purpose of making delays relative to e_a.

Definition 7. $\mathcal{E}_U[\![\ ]\!] : \mathsf{UPA} \longrightarrow \mathsf{UES}$ is defined for $\backslash$, $[\,]$, $+$, $>\!>$, $[>$, and $\mathcal{U}_U()$ as:

$$
\begin{array}{rcl}
\mathcal{E}_U[\![B_1 \,\mathsf{op}\, B_2]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi(B_1 \,\mathsf{op}\, B_2)]\!], \mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle,\ \mathsf{op} \in \{+, [>\} \\
\mathcal{E}_U[\![\mathsf{op}\ B_1]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi(\mathsf{op}\ B_1)]\!], \mathcal{D}_1, \mathcal{T}_1, \mathcal{U}_1 \rangle \text{ for } \mathsf{op} \in \{\, \backslash, [\,] \,\} \\
\mathcal{E}_U[\![\mathcal{U}_U(B_1)]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi(B_1)]\!], \mathcal{D}_1, \mathcal{T}_1, \mathcal{U} \rangle \text{ with } \mathcal{U}(e) = \mathcal{U}_1(e) \ \vee\ (l_1(e) \in U) \\
\mathcal{E}_U[\![B_1 >\!> B_2]\!] & \triangleq & \langle (E_1 \cup E_2, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T}, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle \text{ where} \\
\rightsquigarrow & = & \rightsquigarrow_1 \ \cup\ \rightsquigarrow_2 \ \cup \{\, (e, e') \mid e, e' \in exit(\Psi_1) \ \wedge\ e \neq e' \,\} \\
\mapsto & = & \mapsto_1 \ \cup\ \mapsto_2 \ \cup (\{\, exit(\Psi_1) \,\} \times pin(\Psi_2)) \\
l & = & ((l_1 \cup l_2) \setminus (exit(\Psi_1) \times \{\, \delta \,\})) \cup (exit(\Psi_1) \times \{\, \tau \,\}) \\
\mathcal{D} & = & \mathcal{D}_1 \cup (E_2 \times \{\, 0 \,\}) \\
\mathcal{T} & = & \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{\, ((exit(\Psi_1), e), \mathcal{D}_2(e)) \mid e \in pin(\Psi_2) \,\}.
\end{array}
$$

For op equal to choice or disrupt $\mathcal{E}_U[\![B_1 \ \mathsf{op} \ B_2]\!]$ is the untimed event structure of the corresponding expression in PA, $\mathcal{E}'[\![\Phi(B_1 \ \mathsf{op} \ B_2)]\!]$, where the timings of events and bundles in Ψ_1 and Ψ_2 are unaffected. Similarly, $\mathcal{E}_U[\![\ ]\!]$ is defined for relabelling and hiding. $\mathcal{E}_U[\![\mathcal{U}_U(B)]\!]$ equals $\mathcal{E}'[\![\Phi(B)]\!]$ where each event e with $l_1(e) \in U$ becomes urgent. The events of $\mathcal{E}_U[\![B_1 >> B_2]\!]$ are those in $E_1 \cup E_2$. Bundles are introduced between the successful termination events of Ψ_1 and the positive and initial events in Ψ_2. The reason for introducing bundles to the positive events of Ψ_2 is to make the event delays in Ψ_2 relative to the termination of Ψ_1. This is similar as for timed action-prefix.

Now we consider parallel composition. The events of $[\![B_1 ||_G B_2]\!]$ will be pairs of events of Ψ_1 and Ψ_2, or with one component equal to $*$. Events of Ψ_1 or Ψ_2 that do not need to synchronize are paired with $*$, and an event labelled with an action in G^δ is paired with all events (if any) in the other event structure that are equally labelled (see also the appendix). The delay of an event is the maximum of the delays of its components that are different from $*$. The bundle delay is equal to the maximum of the time associated with the bundles we get by projecting on the i-th components (i=1, 2) of the events in the bundle, if this projection yields a bundle in Ψ_i.

Definition 8. $\mathcal{E}_U[\![\ ]\!] : \mathsf{UPA} \longrightarrow \mathsf{UES}$ is defined for $||_G$ as follows:

$$
\begin{aligned}
\mathcal{E}_U[\![B_1 ||_G B_2]\!] &\triangleq \langle \mathcal{E}'[\![\Phi(B_1 ||_G B_2)]\!], \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle \text{ where} \\
\mathcal{D}((e_1, e_2)) &= max(\mathcal{D}_1(e_1), \mathcal{D}_2(e_2)) \text{ with } \mathcal{D}_i(*) = 0 \\
\mathcal{T}((X, (e_1, e_2))) &= max(Max\{\mathcal{T}_1(X_1, e_1) \mid X_1 \in S_1\}, Max\{\mathcal{T}_2(X_2, e_2) \mid \\
&\quad X_2 \in S_2\}) \text{ with} \\
\mathcal{S}_1 &= \{X_1 \subseteq E_1 \mid X_1 \mapsto_1 e_1 \wedge X = \{(e, e' \in E \mid e \in X_1\}\} \text{ and} \\
\mathcal{S}_2 &= \{X_2 \subseteq E_2 \mid X_2 \mapsto_2 e_2 \wedge X = \{(e, e') \in E \mid e' \in X_2\}\}
\end{aligned}
$$

$$
\mathcal{U}((e_1, e_2)) = \mathcal{U}_1(e_1) \vee \mathcal{U}_2(e_2) \text{ with } \mathcal{U}_i(*) = \text{false}.
$$

EXAMPLE: Figure 3 shows the urgent event structures corresponding to the expressions:

- (a) $((2)\, a\,;\, (3)\, d + (1)\, b\,;\, (2)\, e) \,|||\, (27)\, c$,
- (b) $(14)\, a\,;\, \mathcal{U}_c((2)\, b\,;\, (4)\, c \,||_c\, (7)\, c)$, and
- (c) $((2)\, a\,;\, (5)\, c \,||_c\, (7)\, b\,;\, (1)\, c) \setminus b$.

Case (b) illustrates that in case of action-prefix bundles are introduced to the initial plus the positive events. □

6. Recursion

In this section we consider (process instantiation and) recursion. We assume a behaviour is always considered in the context of a set of process definitions of the form $P := B$ where B is a behaviour possibly containing occurrences of P.

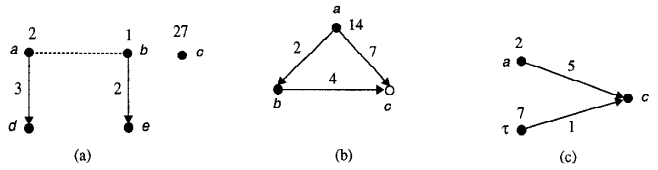


Figure 3. Some example urgent event structure semantics.

$\mathcal{E}_U[\![P]\!]$ for $P := B$ is defined in the following way by using standard fixed point theory [34]. A complete partial order (c.p.o.) $\trianglelefteq$ is defined on urgent event structures with the empty event structure (i.e., $\mathcal{E}_U[\![\mathbf{0}]\!]$) as the least element $\perp$. Then for each definition $P := B$ a function $\mathcal{F}_B$ is defined that substitutes an urgent event structure for each occurrence of P in B, interpreting all operators in B as operators on urgent event structures. $\mathcal{F}_B$ is shown to be continuous, which means that $\mathcal{E}_U[\![P]\!]$ can be defined as the least upper bound (l.u.b.) of the chain (under $\trianglelefteq$) $\perp$, $\mathcal{F}_B(\perp)$, $\mathcal{F}_B(\mathcal{F}_B(\perp)), \ldots$. For this paper we just define the appropriate ordering $\trianglelefteq$, the corresponding l.u.b., and present the main results. Given these ingredients it is rather straightforward to define a function $\mathcal{F}_B$ and prove its continuity. Further details can be found in [20].

Definition 9. Let $\Psi_i = \langle (E_i, \rightsquigarrow_i, \mapsto_i, l_i), \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ for $i = 1, 2$. $\Psi_1 \trianglelefteq \Psi_2$ iff $E_1 \subseteq E_2$, $\rightsquigarrow_1 = \rightsquigarrow_2 \cap (E_1 \times E_1)$, $l_1 = l_2 \restriction E_1$, $\mathcal{D}_1 = \mathcal{D}_2 \restriction E_1, \mathcal{U}_1 = \mathcal{U}_2 \restriction E_1$, and

1. $\mapsto_1 = \{ ((X \cap E_1), e) \mid e \in E_1 \ \wedge \ X \mapsto_2 e \}$, and
2. $\forall e \in E_1 : \mathcal{T}_1((X \cap E_1, e)) = Max\{\mathcal{T}_2(X, e) \mid X \mapsto_2 e\}$.

where $\restriction$ denotes restriction. It is straightforward to verify that $\trianglelefteq$ is a partial order with $\perp = \langle (\varnothing, \varnothing, \varnothing, \varnothing), \varnothing, \varnothing, \varnothing \rangle$ as least element. For conflicts we require that no new conflicts appear in Ψ_2 between events that are already in Ψ_1. Similarly, the first constraint forbids the introduction of bundles in Ψ_2 pointing to events in Ψ_1 for which there exists no projected bundle in Ψ_1. Note that this constraint allows for bundles to grow in such a way that the old bundle set is contained in the new one. The last constraint forces those bundles to get the maximum delay.

The l.u.b. of a chain $\Psi_1 \trianglelefteq \Psi_2 \trianglelefteq \ldots$, denoted $\bigsqcup_i \Psi_i$, can be characterized as follows. For the set of events, conflicts, labelling function, event delays, and urgency predicate we simply take the union of all events, conflicts, labellings, event delays and urgency predicates of the elements in the chain. Since bundles may grow this approach does not apply to the set of bundles. Suppose some Ψ_j has bundle $X_j \mapsto_j e$. According to the definition of $\trianglelefteq$ there is a series of bundles $X_j \mapsto_j e$, $X_{j+1} \mapsto_{j+1} e, \ldots$ satisfying $X_{k+1} \cap E_k = X_k$ for $k \geqslant j$. Then the l.u.b. contains bundle $(\bigcup_n X_{j+n}) \mapsto e$. For $\Psi_1 \trianglelefteq \Psi_2 \trianglelefteq \ldots$:

Definition 10. Let $\bigsqcup_i \Psi_i \triangleq \langle (\bigcup_i E_i, \bigcup_i \rightsquigarrow_i, \mapsto, \bigcup_i l_i), \bigcup_i \mathcal{D}_i, \mathcal{T}, \bigcup_i \mathcal{U}_i \rangle$ with

$$\mapsto = \{ (\bigcup_k X_k, e) \mid \exists\, j : (\forall\, k \geqslant j : X_k \mapsto_k e \ \wedge\ X_{k+1} \cap E_k = X_k) \}$$
$$\mathcal{T} = \{ ((\bigcup_k X_k, e), U_k t_k) \mid \exists\, j : (\forall\, k \geqslant j : X_k \overset{t}{\mapsto} k_k e \ \wedge\ X_{k+1} \cap E_k = X_k) \}.$$

LEMMA 1 *$\bigsqcup_i \Psi_i$ is the least upper bound of chain $\Psi_1 \trianglelefteq \Psi_2 \trianglelefteq \ldots$.*

For the untimed case (cf. [23]) it follows that $\trianglelefteq$ preserves event traces, i.e., $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \Rightarrow T(\mathcal{E}_1) \subseteq T(\mathcal{E}_2)$. This property conforms to the intuition that possible executions of an approximation $\mathcal{E}_{i+1}$ are consistent extensions of possible runs of $\mathcal{E}_i$. A similar property in the urgent case, however, does not hold, since new urgent events in Ψ_{i+1} are allowed [3] and may restrict (or, even prevent) the occurrence of events in Ψ_i. In case urgency does not 'increase' we have a similar result as for the untimed case:

LEMMA 2 $(\Psi_1 \trianglelefteq \Psi_2 \ \wedge\ U(\Psi_1) = U(\Psi_2)) \Rightarrow T_U(\Psi_1) \subseteq T_U(\Psi_2)$.

where $U(\Psi)$ is the set of urgent events in Ψ, i.e., $U(\Psi) \triangleq \{ e \in E \mid \mathcal{U}(e) = \text{true} \}$.

$\trianglelefteq$ corresponds to a weaker notion of trace set inclusion in case the introduction of new urgent events is allowed, but only in such a way that the introduction of conflicts $e \rightsquigarrow e'$, where e' is a new urgent event and e an already existing one, is prohibited. In this case new urgent events will not restrict the occurrence of already existing events, but the 'old' events may be preceded by the new urgent events. E.g., in

$$^{7}\,\mathrm{o}\ b \ \trianglelefteq\ ^{7}\,\mathrm{o}\ b\ ^{2}\,\mathrm{o}\ a$$

$(e_b, 7)$ is not a timed trace of the 'larger' structure, but $(e_a, 2)\,(e_b, 7)$ is: e_b is preceded by a new urgent event, but is not excluded.

As a subsidiary notion we define a weak trace set inclusion relation on sets of timed traces, denoted $\sqsubseteq$. For timed event trace σ and set of events E let $\varepsilon \restriction E \triangleq \varepsilon$ and $((e, a, t)\,\sigma) \restriction E \triangleq (e, a, t)\,(\sigma \restriction E)$ if $e \in E$ and $\sigma \restriction E$ if $e \notin E$. For T_1, T_2 sets of timed event traces let

Definition 11. $T_1 \sqsubseteq T_2 \iff (\forall\, \sigma_1 \in T_1 : (\exists\, \sigma_2 \in T_2 : \sigma_2 \restriction \overline{[\sigma_1]} = \sigma_1))$.

We now have the following result concerning weak trace set inclusion:

LEMMA 3 $(\Psi_1 \trianglelefteq \Psi_2 \ \wedge\ (\rightsquigarrow_2 \cap E_1 \times U(\Psi_2) = \varnothing)) \Rightarrow T_U(\Psi_1) \sqsubseteq T_U(\Psi_2)$.

THEOREM 1 *For $\Psi_1 \trianglelefteq \Psi_2 \trianglelefteq \ldots$ a chain: $T_U(\bigsqcup_i \Psi_i) = \bigcup_i \bigcap_{j \geqslant i} T_U(\Psi_j)$.*

Definition 12. For $P := B$ a process definition let $\mathcal{E}_U[\![\, P \,]\!] \triangleq \bigsqcup_i \mathcal{F}_B^i(\bot)$.

EXAMPLE: As an example of a recursive process definition in UPA we consider $P := \mathcal{U}_a((2)\ a\,;\ P \,|||\, (11)\ b)$. The first approximation is $\bot$, the empty urgent event structure.

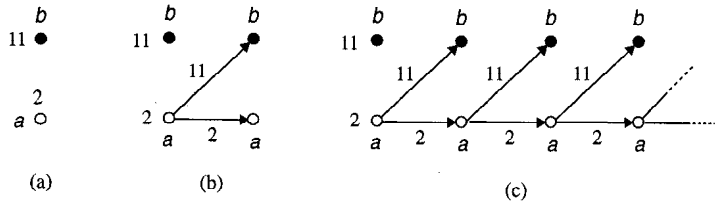


Figure 4. Example of semantics for a recursive process definition in UPA.

The second and third approximation $\mathcal{F}_B(\perp)$ resp. $\mathcal{F}_B^2(\perp)$ are depicted in Figure 4(a) and (b), respectively. Notice that $(e_a, 2)\,(e_b, 11)$ is a timed event trace of (a), but not of (b), since the introduced event labelled a is forced to occur at time 4, so before e_b. By repeated substitution we obtain the urgent event structure of Figure 4(c). □

7. Example: a time-constrained FIFO buffer

In this section we specify a time-constrained first-in first-out (FIFO) buffer. To that extent we extend UPA with a simple form of data; the precise syntax is introduced below. This example is based on [39]; the major difference is that we consider a buffer of infinite length. A simple way to specify a buffer is by using an abstract data type *queue*:

$$Fifo(w : queue) := \sum_{x \in D} \big([w = \langle x \rangle ^\frown w'] \rightarrow rd_x \,;\, Fifo(w') + wr_x \,;\, Fifo(w ^\frown \langle x \rangle)\big)$$

D is a set of data values that can be buffered, wr_x denotes the writing (i.e., insertion) of $x \in D$ into the buffer and rd_x denotes the reading (i.e., removal) of x from the buffer. $\sum$ is a generalized version of the choice operator; $\langle x \rangle$ denotes a singleton queue containing x and $^\frown$ denotes concatenation of queues. $[b] \rightarrow E$ denotes that E can be executed if condition b holds.

The FIFO buffer models a communication network with the following timing constraints [39]: (i) message latency of 2 time units (we do not consider the maximum constraint of [39]); (ii) message input rate set to 1 message per time unit; (iii) message output rate of 1 message per two time units. These time constraints are maintained by the following processes:

$$\begin{aligned}
TD &:= (\sum_{x \in D} wr_x \,;\, (2)\, rd_x) \,|||\, TD \\
Wr &:= \sum_{x \in D} wr_x \,;\, Wr' \text{ where } Wr' := \sum_{x \in D} (1)\, wr_x \,;\, Wr' \\
Rd &:= \sum_{x \in D} rd_x \,;\, Rd' \text{ where } Rd' := \sum_{x \in D} (2)\, rd_x \,;\, Rd' \quad .
\end{aligned}$$

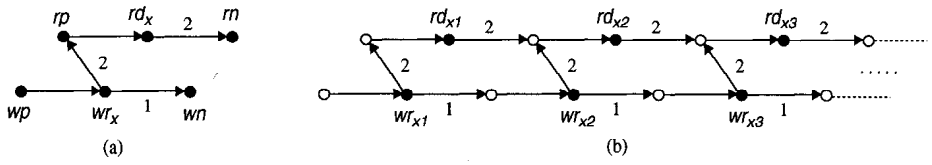


Figure 5. Urgent event structure of a time-constrained FIFO buffer.

The required buffer is obtained by putting these processes in parallel with *Fifo*:

$$Fifo(\langle\,\rangle) \,||\, Rd \,||\, Wr \,||\, TD$$

where $||$ is a shorthand for $||_{\mathsf{Act}}$, i.e., full synchronisation. This specification strongly resembles the timed CSP specification in [39].

A problem with this specification is that it prescribes a mutual exclusion between reading and writing: at any moment one may either choose to read (provided the buffer is not empty) or to write. However, intuitively reading and writing should be to a certain extent independent. If the queue contains one or more elements, it should be possible to read them in parallel with writing new elements. The mutual exclusion constraint is especially unnatural if reading and writing take place at different locations (which is quite common in case of a communication network). We therefore propose a different way of modelling a time-constrained FIFO buffer in which we exploit the use of event structures as a partial order model:

$$\begin{aligned} Cell &:= wp\,;\, \sum_{x\in D} wr_x\,;\, ((1)\; wn \,|||\, (2)\; rp\,;\, rd_x\,;\, (2)\; rn) \\ Chain &:= \mathcal{U}_{\{\,wn,rn\,\}}(Cell \,||_{\{\,wn,rn\,\}}\, Chain[wp := wn, rp := rn]) \\ Buf &:= \mathcal{U}_{\{\,wp,rp\,\}}(Chain) \quad . \end{aligned}$$

The urgent event structures corresponding to the *Cell* and *Buf* processes are depicted in Figure 5(a) and (b), respectively. The labels of the urgent events are omitted for convenience.

Cell describes a buffer cell allowing the writing and reading of a data value. Actions wp and rp ensure that the cell waits before writing (resp. reading); wn and rn indicate the finish of writing and reading and are used in *Chain* to 'start' the next cell. *Chain* puts an unbounded number of cells in parallel using an appropriate renaming function and forces to let write-next and read-next actions to occur urgently. Finally, process *Buf* urges the write-previous and read-previous actions of the front cell.

8. Operational semantics of UPA

Various timed process algebras are known based on an interleaving semantics. In order to compare our noninterleaving approach to these existing approaches and to investigate the 'compatibility' of our proposal with the standard (interleaving) semantics of PA (cf. Table 1) we present an *event-based* operational semantics for UPA. The basic idea is to define a transition system (in the style of [33]) in which we keep track of the (times of) occurrence of actions rather than the actions themselves as usual in structured operational semantics. This results in a *timed event transition system*. The approach is adopted from [23] and is based on [8].

Each occurrence of an action-prefix and $\surd$ is subscripted with an arbitrary but unique event occurrence identifier, denoted by a Greek letter. These occurrence identifiers play the rôle of event names. For parallel composition new event names can be created. If e is an event name of B and e' an event name in B', then possible new names for events in $B \,||_G\, B'$ are $(e, *)$ and $(*, e')$ for unsynchronized events and (e, e') for synchronized events.

We present two sets of SOS -rules that define transition relations $\rightsquigarrow$ and $\longrightarrow$, handling the passage of time and the occurrence of events, respectively. The distinction between these two types of evolvement is inspired by [28, 37] and has been adopted by several others.

The transition relations $\rightsquigarrow$ and $\longrightarrow$ transform a pair $\langle B, t\rangle$, where $B \in$ UPA and $t \in$ Time. $\langle B, t\rangle$ should be interpreted as behaviour B at time t. Usually one starts with $\langle B, 0\rangle$. $\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$ denotes that time can increase from t to t' and in the meantime B changes into B', which is often equal to B, at time t' ($t' \geqslant t$). Thus, time is advanced with an amount of $t'-t$ time units. $\langle B, t\rangle \xrightarrow{(e,a)} \langle B', t\rangle$ means that B at time t performs event e, labelled with action a, and turns into B' (at t).

$\longrightarrow$ and $\rightsquigarrow$ are the smallest relations that are closed under the rules in Tables 2 and 3, respectively. It is assumed that each process instantiation of P is uniquely identified like all occurrences of action-prefix and $\surd$. Different occurrences of the same process instantiation should produce different event transitions. In addition, event transitions cannot be repeated. For $P := (2)\ a_\xi\ ;\ P_\phi$ we first have an event transition with (ξ, a, t) for $t \geqslant 2$; the next time that action a occurs it should be labelled with a label different from ξ. These complications are resolved by using an event renaming operator that prefixes all events in a behaviour with a certain occurrence identifier. $\phi(B)$ is behaviour B where all event identifiers in B are prefixed with ϕ.

The rules for $\longrightarrow$ are straightforward extensions of the standard interleaving semantics of PA. More precisely, by omitting time components from $\langle \ldots \rangle$ and the event and process instantiation identifiers from transitions and expressions we obtain the rules of Table 1. In this sense the presented transition system can be considered to be an *orthogonal* extension of the untimed one.

0 and $\surd$ permit any amount of time to pass, remaining the same behaviour. $(t)\ a_\xi\ ;\ B$ will wait for t time units to become $(0)\ a_\xi\ ;\ B$ after which it either permits any amount of time to pass, remaining the same behaviour, or it may perform event (ξ, a) and behave subsequently like B. Let $t \ominus t'$ denote $max(t-t', 0)$ for $t, t' \in$ Time. For choice, disrupt,

Table 2. SOS-rules for $\longrightarrow$, the event transition relation for UPA.

		$\vdash$	$\langle \surd_\xi, t\rangle \xrightarrow{(\xi,\delta)} \langle \mathbf{0}, t\rangle$
		$\vdash$	$\langle (0)\, a_\xi\, ;\, B, t\rangle \xrightarrow{(\xi,a)} \langle B, t\rangle$
$\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle$		$\vdash$	$\langle B_1 + B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle$
$\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle$		$\vdash$	$\langle B_1 + B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle$
$\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle$	$a \neq \delta$	$\vdash$	$\langle B_1 >> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1' >> B_2, t\rangle$
$\langle B_1, t\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t\rangle$		$\vdash$	$\langle B_1 >> B_2, t\rangle \xrightarrow{(\xi,\tau)} \langle B_2, t\rangle$
$\langle B_1, t\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t\rangle$		$\vdash$	$\langle B_1 [> B_2, t\rangle \xrightarrow{(\xi,\tau)} \langle B_1', t\rangle$
$\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle$		$\vdash$	$\langle B_1 [> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle$
$\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle$	$a \neq \delta$	$\vdash$	$\langle B_1 [> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1' [> B_2, t\rangle$
$\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle$	$a \notin G^\delta$	$\vdash$	$\langle B_1 \,\|\|_G\, B_2, t\rangle \xrightarrow{((\xi,*),a)} \langle B_1' \,\|\|_G\, B_2, t\rangle$
$\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle$	$a \notin G^\delta$	$\vdash$	$\langle B_1 \,\|\|_G\, B_2, t\rangle \xrightarrow{((*,\xi),a)} \langle B_1 \,\|\|_G\, B_2', t\rangle$
$\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle \wedge \langle B_2, t\rangle \xrightarrow{(\psi,a)} \langle B_2', t\rangle$	$a \in G^\delta$	$\vdash$	$\langle B_1 \,\|\|_G\, B_2, t\rangle \xrightarrow{((\xi,\psi),a)} \langle B_1' \,\|\|_G\, B_2', t\rangle$
$\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle$	$a \notin G$	$\vdash$	$\langle B \setminus G, t\rangle \xrightarrow{(\xi,a)} \langle B' \setminus G, t\rangle$
$\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle$	$a \in G$	$\vdash$	$\langle B \setminus G, t\rangle \xrightarrow{(\xi,\tau)} \langle B' \setminus G, t\rangle$
$\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle$		$\vdash$	$\langle B[H], t\rangle \xrightarrow{(\xi,H(a))} \langle B'[H], t\rangle$
$\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle$		$\vdash$	$\langle \mathcal{U}_U(B), t\rangle \xrightarrow{(\xi,a)} \langle \mathcal{U}_U(B'), t\rangle$
$\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle$	$P := B$	$\vdash$	$\langle P_\phi, t\rangle \xrightarrow{(\phi\xi,a)} \langle \phi(B'), t\rangle$
$\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle$		$\vdash$	$\langle \phi(B), t\rangle \xrightarrow{(\phi\xi,a)} \langle \phi(B'), t\rangle$

and parallel composition time passes for the composite behaviour if all components can do so. For enabling the first behaviour determines whether time can advance. The rules for hiding, relabelling and process instantiation are straightforward. If B permits time to pass with some amount, then $\mathcal{U}_U(B)$ is able to do the same provided that there is no urgent action in U that can be performed by B at any time earlier. Thus, the effect of the urgency operator is to prevent the passage of time as an alternative to the occurrence of an action in the urgent set U.

$d_{min}(a, B)$ denotes for initial action a in B the minimal time at which it can appear. The interpretation of $d_{min}(a, B) = \infty$ is that B is not able to perform an a action initially.

Definition 13. For $a \in \mathsf{Act} \cup \{\, \tau, \delta \,\}$ and $B \in \mathsf{UPA}$, function d_{min} is defined as:

$$
\begin{aligned}
d_{min}(a, \mathbf{0}) &\triangleq \infty \\
d_{min}(a, \surd) &\triangleq \begin{cases} \infty & \text{if } a \neq \delta \\ 0 & \text{if } a = \delta \end{cases} \\
d_{min}(a, (t)\, b\, ;\, B) &\triangleq \begin{cases} \infty & \text{if } a \neq b \\ t & \text{if } a = b \end{cases} \\
d_{min}(a, B_1 + B_2) &\triangleq min(d_{min}(a, B_1), d_{min}(a, B_2)) \\
d_{min}(a, B_1 >> B_2) &\triangleq \begin{cases} \infty & \text{if } a = \delta \\ d_{min}(a, B_1) & \text{if } a \notin \{\, \tau, \delta \,\} \\ min(d_{min}(a, B_1), d_{min}(\delta, B_1)) & \text{if } a = \tau \end{cases}
\end{aligned}
$$

Table 3. SOS-rules for $\rightsquigarrow$, the time advancing transition relation for UPA.

	$t' \geqslant t$	$\vdash$	$\langle \mathbf{0}, t\rangle \rightsquigarrow \langle \mathbf{0}, t'\rangle$
	$t' \geqslant t$	$\vdash$	$\langle \surd_\xi, t\rangle \rightsquigarrow \langle \surd_\xi, t'\rangle$
	$t'' \geqslant t$	$\vdash$	$\langle (t')\, a_\xi \,;\, B, t\rangle \rightsquigarrow \langle (t' \ominus (t''-t))\, a_\xi \,;\, B, t''\rangle$
$\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge$	$\langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle$	$\vdash$	$\langle B_1 + B_2, t\rangle \rightsquigarrow \langle B_1' + B_2', t'\rangle$
	$\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle$	$\vdash$	$\langle B_1 >> B_2, t\rangle \rightsquigarrow \langle B_1' >> B_2, t'\rangle$
$\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge$	$\langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle$	$\vdash$	$\langle B_1 [> B_2, t\rangle \rightsquigarrow \langle B_1' [> B_2', t'\rangle$
$\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge$	$\langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle$	$\vdash$	$\langle B_1 \,\|\|_G\, B_2, t\rangle \rightsquigarrow \langle B_1' \,\|\|_G\, B_2', t'\rangle$
	$\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$	$\vdash$	$\langle B \setminus G, t\rangle \rightsquigarrow \langle B' \setminus G, t'\rangle$
	$\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$	$\vdash$	$\langle B[H], t\rangle \rightsquigarrow \langle B'[H], t'\rangle$
$\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$	$\forall\, a \in U : t'-t \leqslant d_{min}(a, B)$	$\vdash$	$\langle \mathcal{U}_U(B), t\rangle \rightsquigarrow \langle \mathcal{U}_U(B'), t'\rangle$
	$\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle \quad P := B$	$\vdash$	$\langle P_\phi, t\rangle \rightsquigarrow \langle \phi(B'), t'\rangle$
	$\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$	$\vdash$	$\langle \phi(B), t\rangle \rightsquigarrow \langle \phi(B'), t'\rangle$

$$
\begin{aligned}
d_{min}(a, B_1 \,[>\, B_2) &\triangleq min(d_{min}(a, B_1), d_{min}(a, B_2)) \\
d_{min}(a, B_1 \,||_G\, B_2) &\triangleq \begin{cases} min(d_{min}(a, B_1), d_{min}(a, B_2)) & \text{if } a \notin G^\delta \\ max(d_{min}(a, B_1), d_{min}(a, B_2)) & \text{if } a \in G^\delta \end{cases} \\
d_{min}(a, B \setminus G) &\triangleq \begin{cases} Min\{\, d_{min}(b, B) \mid b \in G^\tau \,\} & \text{if } a = \tau \\ \infty & \text{if } a \in G \\ d_{min}(a, B) & \text{if } a \notin G^\tau \end{cases} \\
d_{min}(a, B[H]) &\triangleq Min\{\, d_{min}(b, B) \mid a = H(b) \,\} \\
d_{min}(a, \mathcal{U}_U(B)) &\triangleq d_{min}(a, B) \\
d_{min}(a, P) &\triangleq d_{min}(a, B) \text{ for } P := B.
\end{aligned}
$$

Here it is assumed that *min*, *max* and their generalisations over sets of events are defined on Time $\cup\, \{\infty\}$ in the obvious way. E.g., $min(t, \infty) \triangleq t$ and $max(t, \infty) \triangleq \infty$. In order to let d_{min} be well defined we require process instantiations to occur in a weakly guarded way (i.e., they should become guarded after a finite number of substitutions of bodies for their process names). The correctness of the function d_{min} is addressed in the next section.

In order to constrain the passage of time in the inference rules for **time** $a(t_1, t_2)$ **in** B, [7] use a function $age(a, B)$ which determines the maximal (rather than the minimal) time at which B can perform initial action a. The correctness of *age* is, however, not addressed by [7].

9. Consistency of denotational and operational semantics

In this section we investigate the relationship between the causality-based and operational semantics of UPA. For convenience we introduce the transition relation $\longrightarrow_*$.

Definition 14. $\langle B,t\rangle \xrightarrow{(e,a,t')}_* \langle B',t'\rangle$ if and only if there exists a behaviour B'' such that $\langle B,t\rangle \rightsquigarrow \langle B'',t'\rangle$ and $\langle B'',t'\rangle \xrightarrow{(e,a)} \langle B',t'\rangle$.

Using the relation $\longrightarrow_*$ the notion of timed event trace and a trace derivation relation $\xrightarrow{\sigma}_*$ can be defined in the usual way. We summarize the following results, where for $B \in \mathsf{UPA}$ the set $U(B)$ denotes the set of urgent actions in B. $U(B)$ can easily be defined by induction on the structure of B.

THEOREM 2 *For all* $B, B' \in \mathsf{UPA}$, $t, t' \in \mathsf{Time}$ *and* $a \in \mathsf{Act} \cup \{\tau, \delta\}$ *we have*

1. $\langle B,t'\rangle \xrightarrow{(e,a,t)}_* \;\Rightarrow\; t \geqslant t' + d_{min}(a,B)$.
2. $\forall\, b \in U(B) : \langle B,t'\rangle \xrightarrow{(e,b,t)}_* \;\Rightarrow\; t = t' + d_{min}(b,B)$.
3. $d_{min}(a,B) = \infty \Leftrightarrow \left(\forall\, t,t' : \langle B,t\rangle \xrightarrow{(e,a,t')} \!\!\!\!/\;\;_*\right)$.
4. $\langle B,t\rangle \xrightarrow{(e,a,t')}_* \;\Rightarrow\; t' \leqslant t + Min\{\, d_{min}(b,B) \mid b \in U(B)\,\}$.
5. $\left(\langle B,t\rangle \xrightarrow{(e,a,t_a)}_* \;\wedge\; \langle B,t\rangle \rightsquigarrow \langle B',t'\rangle\right) \;\Rightarrow\; d_{min}(a,B') = d_{min}(a,B) \ominus (t'-t)$.

1. expresses that the time determined by $d_{min}(a,B)$ corresponds to the earliest moment at which initial action a can be performed by B. 2. says that urgent actions in B can only happen as soon as they are enabled. If $d_{min}(a,B) = \infty$ then B is not able to perform a initially. This is stated in 3. 4. states that actions can only be performed by B provided there is no urgent action in B that could occur earlier. Finally, 5. shows the relation between d_{min} and $\rightsquigarrow$.

We now consider the well-known properties time determinism, action persistency, and time additivity [30] for UPA.

THEOREM 3 *For all* $B, B', B'' \in \mathsf{UPA}$, $t, t', t'' \in \mathsf{Time}$ *we have*

1. *Time determinism:* $\left(\langle B,t\rangle \rightsquigarrow \langle B',t'\rangle \;\wedge\; \langle B,t\rangle \rightsquigarrow \langle B'',t'\rangle\right) \;\Rightarrow\; B' = B''$.
2. *Action persistency:* $\left(\langle B,t\rangle \xrightarrow{(e,a)} \;\wedge\; \langle B,t\rangle \rightsquigarrow \langle B',t'\rangle\right) \;\Rightarrow\; \langle B',t'\rangle \xrightarrow{(e,a)}$.
3. *Time additivity*[4] *:* $\langle B,t\rangle \rightsquigarrow \langle B', t{+}(t'{+}t'')\rangle$ *iff*
   $(\exists\, B'' : \langle B,t\rangle \rightsquigarrow \langle B'', t{+}t'\rangle \rightsquigarrow \langle B', t{+}(t'{+}t'')\rangle)$.

The proofs of the above theorems are by induction on the structure of B. They are rather straightforward, but somewhat tedious, and are omitted here; see [20].

Since the transition system under $\longrightarrow_*$ is deterministic, this transition system can be represented by its set of timed event traces $\mathcal{T}_U[\![\, B \,]\!]$. This set can be characterized in a

denotational way, and subsequently proven to coincide with the set of timed event traces of the corresponding urgent event structure $\mathcal{E}_U[\![B]\!]$. We thus have the following consistency result:

THEOREM 4 $\forall B \in \mathsf{UPA} : T_U(\mathcal{E}_U[\![B]\!]) = \mathcal{T}_U[\![B]\!]$.

For finite behaviours the proof of this theorem is quite lengthy (but not so complicated) and omitted here for space reasons; see [20]. The main crux is to characterize correctly the timed traces of $+$, $[>$, and $\mathcal{U}_U()$ in a denotational way and to prove that this characterisation coincides with the timed event traces of the corresponding urgent event structure. For recursive behaviours the proof is not so trivial, and we will deal with this case in more detail. Here we assume that process instantiations are guarded; this is not a further restriction, since any weakly guarded expression can be turned into a guarded one by some substitutions.

Definition 15. For $B, B' \in \mathsf{UPA}$ and process instantiation P, $B'[P := B]$, is defined as

$$
\begin{array}{rcl}
\mathbf{0}[P := B] & \triangleq & \mathbf{0} \\
\surd[P := B] & \triangleq & \surd \\
(\mathsf{op}\ B_1)[P := B] & \triangleq & \mathsf{op}\ (B_1[P := B]) \text{ for } \mathsf{op} \in \{\, a\,;, \backslash, [\,], \mathcal{U}_U()\,\} \\
(B_1\ \mathsf{op}\ B_2)[P := B] & \triangleq & (B_1[P := B])\ \mathsf{op}\ (B_2[P := B]) \text{ for } \mathsf{op} \in \{\, +, >>, [>, \,||\,\} \\
Q[P := B] & \triangleq & \begin{cases} \phi(B) & \text{if } Q = P_\phi \\ Q & \text{if } Q \neq P. \end{cases}
\end{array}
$$

$B'[P := B]$ denotes behaviour B' where all occurrences of P_ϕ in B' are replaced with $\phi(B)$. As a next subsidiary notion we define the unfoldings of P. The n-th approximation of P is defined as the n-th unfolding where each occurrence of P is replaced by $\mathbf{0}$.

Definition 16. For $P := B$ the n-th *unfolding* of P, denoted $\hat{P}^n$, is defined as: $\hat{P}^0 \triangleq P$ and $\hat{P}^{n+1} \triangleq B[P := \hat{P}^n]$. The n-th *approximation* of P, denoted P^n, is defined as $P^n \triangleq \hat{P}^n[P := \mathbf{0}]$.

The following result can easily be proven by induction on n. We have that the set of timed event traces of P is equal to that of its n-th unfolding.

LEMMA 4 $\forall n \geqslant 0 : \mathcal{T}_U[\![P]\!] = \mathcal{T}_U[\![\hat{P}^n]\!]$.

If $B \xrightarrow{(e,a,t)}_* B'$ and B is guarded then this transition can be derived without applying one of the transition rules for recursive process behaviours (cf. Tables 2 and 3) due to the guardedness of B. But then, the process instantiations occurring in B may be replaced by some arbitrary expression X without prohibiting this transition.

LEMMA 5 *Let $B \in \mathsf{UPA}$ such that B is guarded. Then for arbitrary $X \in \mathsf{UPA}$ and process identifier P we have:* $B \xrightarrow{(e,a,t)}_* B' \Rightarrow B[P := X] \xrightarrow{(e,a,t)}_* B'[P := X]$.

The following lemma is based on the intuition that traces of length at most n can involve at most n unfoldings of process instantiations. More precisely, it states that if σ is a timed trace of B' where all occurrences of P are replaced by its n-th unfolding $\hat{P}^n$ and $\mid \sigma \mid \leqslant n$, then P may be replaced in the resulting term by an arbitrary term X while preserving that σ is a timed trace.

LEMMA 6 *Let $B' \in$ UPA possibly containing unguarded occurrences of P, for $P := B$ and B guarded. Then for arbitrary term $X \in$ UPA, for $n \geqslant 0$:*

$$\Big(\sigma \in \mathcal{T}_U[\![\, B'[P := \hat{P}^n]\,]\!] \;\wedge\; \mid \sigma \mid \leqslant n\Big) \;\Rightarrow\; \sigma \in \mathcal{T}_U[\![\, B'[P := \hat{P}^n[P := X]]\,]\!].$$

Proof: By induction on n.
Base: for n=0 the lemma trivially holds since ε is a trace of any behaviour.
Induction step: Assume the lemma holds for n=k; consider k+1. First we derive

$$\begin{array}{ll} & B'[P := \hat{P}^{k+1}] \\ = & \{\text{ definition of unfolding }\} \\ & B'[P := B[P := \hat{P}^k]] \\ = & \{\text{ substitution property }\} \\ & B'[P := B][P := \hat{P}^k] \quad . \end{array}$$

In a similar way we can derive that

$$B'[P := \hat{P}^{k+1}[P := X]] = B'[P := B][P := \hat{P}^k[P := X]] \quad . \tag{1}$$

Now assume $\sigma \in \mathcal{T}_U[\![\, B'[P := B][P := \hat{P}^k]\,]\!]$ with $\sigma = (e, a, t)\,\sigma'$ and $\mid \sigma' \mid = k$. Then:

$$\begin{array}{ll} & \sigma \in \mathcal{T}_U[\![\, B'[P := \hat{P}^{k+1}[P := X]]\,]\!] \\ \Leftrightarrow & \{\,(1)\,\} \\ & \sigma \in \mathcal{T}_U[\![\, B'[P := B][P := \hat{P}^k[P := X]]\,]\!] \\ \Leftrightarrow & \{\,\sigma = (e, a, t)\,\sigma';\ B \text{ is guarded}\,\} \\ & \quad B'[P := B][P := \hat{P}^k[P := X]] \xrightarrow{(e,a,t)}_* B''[P := \hat{P}^k[P := X]] \\ & \wedge\ \sigma' \in \mathcal{T}_U[\![\, B''[P := \hat{P}^k[P := X]]\,]\!] \\ \Leftarrow & \{\, B'[P := B] \text{ is guarded (since } B \text{ is); Lemma 5 }\} \\ & B'[P := B][P := \hat{P}^k] \xrightarrow{(e,a,t)}_* B''[P := \hat{P}^k] \;\wedge\; \sigma' \in \mathcal{T}_U[\![\, B''[P := \hat{P}^k[P := X]]\,]\!] \\ \Leftarrow & \{\text{ induction hypothesis }\} \\ & B'[P := B][P := \hat{P}^k] \xrightarrow{(e,a,t)}_* B''[P := \hat{P}^k] \;\wedge\; \sigma' \in \mathcal{T}_U[\![\, B''[P := \hat{P}^k]\,]\!] \\ \Leftrightarrow & \{\,\sigma = (e, a, t)\,\sigma';\ B \text{ is guarded}\,\} \\ & \sigma \in \mathcal{T}_U[\![\, B'[P := B][P := \hat{P}^k]\,]\!] \quad . \end{array}$$

■

THEOREM 5 *For $P := B$; and guarded B we have $\mathcal{T}_U[\![\, P\,]\!] = \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}_U[\![\, P^j\,]\!]$.*

Proof: The proof for the '$\supseteq$' part follows from the fact that if a trace belongs to almost all approximations from the i-th approximation on, i.e., $\sigma \in \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}_U[\![P^j]\!]$, then it belongs to the limit as well. The proof for '$\subseteq$' follows.

'$\subseteq$':

$\sigma \in \mathcal{T}_U[\![P]\!] \ \wedge \ |\, \sigma \,| \leqslant n$

$\Leftrightarrow$ { definition of substitution }

$\sigma \in \mathcal{T}_U[\![P[P := P]]\!] \ \wedge \ |\, \sigma \,| \leqslant n$

$\Leftrightarrow$ { Lemma 4 }

$\sigma \in \mathcal{T}_U[\![P[P := \hat{P}^n]]\!] \ \wedge \ |\, \sigma \,| \leqslant n$

$\Rightarrow$ { Lemma 6 }

$\sigma \in \mathcal{T}_U[\![P[P := \hat{P}^n[P := \mathbf{0}]]]\!]$

$\Leftrightarrow$ { definition of approximation }

$\sigma \in \mathcal{T}_U[\![P[P := P^n]]\!]$

$\Leftrightarrow$ { definition of substitution }

$\sigma \in \mathcal{T}_U[\![P^n]\!]$.

Since this holds for all n it follows that $\sigma \in \mathcal{T}_U[\![P]\!] \ \Rightarrow \ \sigma \in \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}_U[\![P^j]\!]$. ■

Then we have the following consistency result between the denotational semantics in terms of urgent event structures and the event-based operational semantics.

THEOREM 6 *For $P := B$ and guarded B we have $T_U(\mathcal{E}_U[\![P]\!]) = \mathcal{T}_U[\![P]\!]$.*

Proof: Straightforward by the definition of $\mathcal{E}_U[\![P]\!]$, Theorems 1 and 5, and the consistency of the denotational and operational semantics for finite behaviours. ■

10. Related work

Timed extensions of partial-order models have received scant attention in the literature. Extensions of pomsets [12], configurations [26], event structures [29], posets [18], {and, or}-automata [16] and higher-dimensional automata [15] are known to us. Our approach resembles that of [14]. Fidge proposes a real-time extension of causal trees and uses this model to provide a semantics to a timed variant of CCS. Each event e in a causal tree has a set of backward pointers to each event on which e causally depends. Time constraints are expressed by associating a set of relative times to events. The relative delays I state that an event can only occur at t time units (for some $t \in I$) after the time at which *all* its causally preceding events occurred (if any). Synchronisation can only occur if both participants are willing to engage in the interaction at the same time instant; if not, the synchronisation will not take place. The main distinctions with our approach are that we use a branching time model, associate time to bundles and events, and we allow for the co-existence of urgent and non-urgent events. Due to the adjustments of backward pointers the SOS-rules in [14] are somewhat complicated and the relation with the standard rules for CCS is not so clear.

Recently, Bowman [9] reported on a timed LOTOS extension with a semantics using a timed variant of bundle event structures. His approach does not include $\lfloor>$, uses a start event for recording absolute delays and forces all internal events to be urgent. [9] does not report on the consistency with an operational semantics.

For the untimed case several approaches exist that relate a causality-based semantics to an interleaving one [4, 8, 24]. These investigations differ from our work in particular in the causality-based model, the language at hand, and the type of consistency relation between the two types of semantics. [4, 24] prove the consistency between an operational semantics for Theoretical CSP (TCSP) and a compositional true concurrency semantics based on labelled prime event structures. They show that the 'interleaved view' of the event structure semantics—obtained by considering remainders of event structures after the execution of a single event—is (weak) bisimilar to the operational semantics of TCSP. [13] proposed an approach to prove the consistency of an operational noninterleaving semantics of CCS (with guarded recursion) and a denotational one based on labelled prime event structures. From the operational semantics an occurrence net is derived which is shown—using the well-known connection between this class of nets and event structures [31]—to be equal to the event structure obtained in the denotational way.

11. Conclusion and discussion

This paper introduced a temporal process algebra UPA with just two timed features: a simple delay function and an urgency operator. A novel timed enhancement of event structures is used to provide a causality-based semantics in a compositional way. This semantics is an upward compatible semantics of the untimed noninterleaving semantics of LOTOS: omitting the time delays from events and bundles, and ignoring the urgency indications, leads to the untimed partial-order semantics. This supports the design approach where untimed behaviours are refined into corresponding timed behaviours.

In addition, an operational semantics is given inspired by the separation of the passage of time (relation $\rightsquigarrow$) and the occurrence of actions (relation $\longrightarrow$). It turns out that the transition system for $\longrightarrow$ is identical to the standard (interleaving) semantics of LOTOS when omitting time and event annotations. Thus, time is added in a completely orthogonal way. This provides a solid basis for the use of both partial-order and interleaving semantics in a design approach where one starts with an observational (interleaving) description of the system's behaviour and ends with a (partial-order) prescription for the system's realisation.

The presented operational semantics of UPA turns out to be very close to the proposal of [7]. The main difference with this proposal is the treatment of synchronisation on urgent actions—[7] allows them at the prize of introducing time deadlocks (yielding an execution-stop for the entire system), whereas our proposal avoids them. Since the operational and denotational semantics of UPA are consistent in the sense that identical sets of timed event traces are generated we consider the aforementioned characteristics to provide evidence for the adequacy of our timed causality-based model.

Acknowledgments

The authors would like to thank Arend Rensink for his suggestions and discussion.

An extended abstract of a preliminary version of this paper appeared as J-P. Katoen, D. Latella, R. Langerak, E. Brinksma and T. Bolognesi. A consistent causality-based view on a timed process algebra. In *Proceedings 3rd Amast Workshop on Real-Time Systems*, pages 212–227. Salt Lake City, Utah, USA, March 1996.

The work in this paper is partially funded by C.N.R. - Progetto Bilaterale: Estensioni probabilistiche e temporali dell'algebra di processi LOTOS basate su strutture di eventi, per la specifica e analisi quantitative di sistemi distribuiti, by C.N.R. - Progetto Coordinato: Strumenti per la specifica e verifica di proprieta' critiche di sistemi concorrenti e distribuiti, and by the EU as part of the ESPRIT BRA project 6021: Building Correct Reactive Systems (REACT).

Appendix Denotational semantics of PA

Here we provide the full definition of the causality-based semantics of PA. The initial events and successful termination events of an event structure are: $init(\mathcal{E}) \triangleq \{ e \in E \mid \neg(\exists X \subseteq E : X \mapsto e) \}$ and $exit(\mathcal{E}) \triangleq \{ e \in E \mid l(e) = \delta \}$. We suppose there is an infinite universe E_U of events. In the rest of this section let $\mathcal{E}[\![B_i]\!] = \mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$, for i=1, 2 with $E_1 \cap E_2 = \varnothing$. (If $E_1 \cap E_2 \neq \varnothing$ then a suitable event renaming can be applied extended to $\rightsquigarrow$, $\mapsto$, and l.)

The semantics of **0** and $\surd$ is self-explanatory. In $\mathcal{E}[\![a\,;\, B_1]\!]$ a bundle is introduced from the new event e_a (labelled a) to all initial events in $\mathcal{E}_1$ as e_a causally precedes these events. $\mathcal{E}[\![B_1 + B_2]\!]$ is equal to the union of $\mathcal{E}_1$ and $\mathcal{E}_2$ extended with mutual conflicts between all initial events of $\mathcal{E}_1$ and $\mathcal{E}_2$ such that in the resulting structure only either B_1 or B_2 can happen.

$\mathcal{E}[\![B_1 \setminus G]\!]$ is identical to $\mathcal{E}_1$ except that events labelled with a label in G are now labelled with τ turning those events into internal ones. $\mathcal{E}[\![B_1[H]]\!]$ is defined similarly where events are relabelled according to H ($\circ$ denotes usual function composition).

$\mathcal{E}[\![B_1 >> B_2]\!]$ is equal to the union of $\mathcal{E}_1$ and $\mathcal{E}_2$ where bundles are introduced from the successful termination events of $\mathcal{E}_1$ to the initial events of $\mathcal{E}_2$. (To create bundles, mutual conflicts are introduced between the successful termination events of $\mathcal{E}_1$.) This corresponds with the fact that these initial events can only occur if B_1 has successfully terminated. The successful termination events of $\mathcal{E}_1$ are relabelled into internal events.

$\mathcal{E}[\![B_1 [> B_2]\!]$ is equal to the union of $\mathcal{E}_1$ with $\mathcal{E}_2$ extended with some additional asymmetric conflicts. First, each event in $\mathcal{E}_1$ may be disabled by an initial event of $\mathcal{E}_2$. This models that B_1 is disrupted once an initial event of B_2 happens. In addition, after the occurrence of a successful termination event in $\mathcal{E}_1$ no initial event of $\mathcal{E}_2$ can happen anymore.

The events of $\mathcal{E}[\![B_1 ||_G B_2]\!]$ are constructed in the following way: an event e of $\mathcal{E}_1$ or $\mathcal{E}_2$ that does not need to synchronize is paired with the auxiliary symbol $*$, and an event which

is labelled with an action in G^δ is paired with all events (if any) in the other process that are equally labelled. Thus events are pairs of events of $\mathcal{E}_1$ and $\mathcal{E}_2$, or with one component equal to $*$. Two events are now put in conflict if any of their components are in conflict, or if different events have a common component different from $*$ (such events appear if two or more events in one process synchronize with the same event in the other process). A bundle is introduced such that if we take the projection on the i-th component (i=1, 2) of all events in the bundle we obtain a bundle in $\mathcal{E}[\![B_i]\!]$.

For $G \subseteq$ Act, $E_i^s \triangleq \{ e \in E_i \mid l_i(e) \in G^\delta \}$ is the set of *synchronisation* events and $E_i^f \triangleq E_i \setminus E_i^s$ the set of *non-synchronizing* events.

Definition 17. $\mathcal{E}[\![\ ]\!] : \mathsf{PA} \longrightarrow \mathsf{EBES}$ is defined as follows:

$$
\begin{array}{rcl}
\mathcal{E}[\![\mathbf{0}]\!] & \triangleq & (\varnothing, \varnothing, \varnothing, \varnothing) \\
\mathcal{E}[\![\surd]\!] & \triangleq & (\{ e_\delta \}, \varnothing, \varnothing, \{ (e_\delta, \delta) \}) \text{ for some } e_\delta \in E_U \\
\mathcal{E}[\![a\,;\, B_1]\!] & \triangleq & (E, \rightsquigarrow_1, \mapsto, l_1 \cup \{ (e_a, a) \}) \text{ where} \\
E & = & E_1 \cup \{ e_a \} \text{ for some } e_a \in E_U \setminus E_1 \\
\mapsto & = & \mapsto_1 \ \cup\ (\{ \{ e_a \} \} \times init(\mathcal{E}_1)) \\
\mathcal{E}[\![B_1 + B_2]\!] & \triangleq & (E_1 \cup E_2, \rightsquigarrow, \mapsto_1 \cup \mapsto_2, l_1 \cup l_2) \text{ where} \\
\rightsquigarrow & = & \rightsquigarrow_1 \cup \rightsquigarrow_2 \cup (init(\mathcal{E}_1) \times init(\mathcal{E}_2)) \cup (init(\mathcal{E}_2) \times init(\mathcal{E}_1)) \\
\mathcal{E}[\![B_1 \setminus G]\!] & \triangleq & (E_1, \rightsquigarrow_1, \mapsto_1, l) \text{ where} \\
 & & (l_1(e) \in G \ \Rightarrow\ l(e) = \tau) \ \wedge\ (l_1(e) \notin G \ \Rightarrow\ l(e) = l_1(e)) \\
\mathcal{E}[\![B_1[H]]\!] & \triangleq & (E_1, \rightsquigarrow_1, \mapsto_1, H \circ l_1) \\
\mathcal{E}[\![B_1 >> B_2]\!] & \triangleq & (E_1 \cup E_2, \rightsquigarrow, \mapsto, l) \text{ where} \\
\rightsquigarrow & = & \rightsquigarrow_1 \ \cup\ \rightsquigarrow_2 \ \cup \{ (e, e') \mid e, e' \in exit(\mathcal{E}_1) \ \wedge\ e \neq e' \} \\
\mapsto & = & \mapsto_1 \ \cup\ \mapsto_2 \ \cup (\{ exit(\mathcal{E}_1) \} \times init(\mathcal{E}_2)) \\
l & = & ((l_1 \cup l_2) \setminus (exit(\mathcal{E}_1) \times \{ \delta \})) \cup (exit(\mathcal{E}_1) \times \{ \tau \}) \\
\mathcal{E}[\![B_1 [> B_2]\!] & \triangleq & (E_1 \cup E_2, \rightsquigarrow, \mapsto_1 \ \cup\ \mapsto_2, l_1 \cup l_2) \text{ where} \\
\rightsquigarrow & = & \rightsquigarrow_1 \ \cup\ \rightsquigarrow_2 \ \cup (E_1 \times init(\mathcal{E}_2)) \cup (init(\mathcal{E}_2) \times exit(\mathcal{E}_1)) \\
\mathcal{E}[\![B_1 ||_G B_2]\!] & \triangleq & (E, \rightsquigarrow, \mapsto, l) \text{ where} \\
E & = & (E_1^f \times \{ * \}) \cup (\{ * \} \times E_2^f) \cup \\
 & & \{ (e_1, e_2) \in E_1^s \times E_2^s \mid l_1(e_1) = l_2(e_2) \} \\
(e_1, e_2) \rightsquigarrow (e_1', e_2') & \Leftrightarrow & (e_1 \rightsquigarrow_1 e_1') \ \vee\ (e_2 \rightsquigarrow_2 e_2') \ \vee \\
 & & (e_1 = e_1' \neq * \ \wedge\ e_2 \neq e_2') \vee (e_2 = e_2' \neq * \ \wedge\ e_1 \neq e_1') \\
X \mapsto (e_1, e_2) & \Leftrightarrow & (\exists\, X_1 \subseteq E_1 : X_1 \mapsto_1 e_1 \wedge X = \{ (e, e') \in E \mid e \in X_1 \}) \\
 & & \vee (\exists\, X_2 \subseteq E_2 : X_2 \mapsto_2 e_2 \wedge X = \{ (e, e') \in E \mid e' \in X_2 \}) \\
l((e_1, e_2)) & = & \text{if } e_1 = * \text{ then } l_2(e_2) \text{ else } l_1(e_1).
\end{array}
$$

For technical reasons, in this paper we use a slightly different version of $\mathcal{E}[\![\ ]\!]$, denoted $\mathcal{E}'[\![\ ]\!]$. The only difference with $\mathcal{E}[\![\ ]\!]$ is that for action-prefix $\mathcal{E}'[\![\ ]\!]$ introduces not only bundles from e_a to the initial events of Ψ_1 (as above), but to some more events in Ψ_1. Similarly, for enabling $\mathcal{E}'[\![\ ]\!]$ introduces bundles from the successful termination events of Ψ_1 to some more events in Ψ_2. The precise set of events depends on the timing constructs as introduced in this paper, cf. Section 5. These additional bundles do not pose any problems, since $T(\mathcal{E}[\![B]\!]) = T(\mathcal{E}'[\![B]\!])$, for all $B \in \mathsf{PA}$, see [20].

Notes

1. The terminology 'asymmetric' is adopted from [23, 32].
2. Alternatively, we could explicitly model the start of the system by some fictitious event, ω say. Then the time associated to event e can be considered as the time associated to the bundle pointing from ω to e. We do not consider the introduction of such event ω since the definitions become more complex—ω has to be treated differently than 'normal' events—and proof obligations become more severe—e.g., one has to prove that bundles $X \mapsto e$ satisfy $X = \{\, \omega \,\}$, or $\omega \notin X$ and $e \neq \omega$. Such approach has been taken by [9].
3. And this should be allowed in order to guarantee the monotonicity of $+$, $||_G$, and so on, interpreted on urgent event structures (as used in $\mathcal{F}_B$) with respect to $\trianglelefteq$
4. For timed I/O-automata [25] a stronger notion is adopted that says that there must be a *trajectory* of consistent states through the interval $[t, t']$. Since our timed transition system satisfies the image-finiteness condition (that is, for any B and t' there are at most finitely many B' such that $\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$) it follows from [19] that our model also satisfies this stronger trajectory condition.

References

1. S. Abramski. Observation equivalence as a testing equivalence. *Theoretical Computer Science*, 53:225–241, 1987.
2. J.C.M. Baeten and J.W. Klop, editors. *Concur '90: Theories of Concurrency – Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.
3. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
4. C. Baier and M.E. Majster-Cederbaum. The connection between an event structure semantics and an operational semantics for TCSP. *Acta Informatica*, 31:81–104, 1994.
5. T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25-59, 1987.
6. T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. In de Bakker et al., editors, *Real-time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 124-148, 1992.
7. T. Bolognesi, F. Lucidi, and S. Trigila. Converging towards a timed LOTOS standard. *Computer Standards & Interfaces*, 16:87-118, 1994.
8. G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. *Information and Computation*, 114:247-314, 1994.
9. H. Bowman. A true concurrency approach to time extended LOTOS (revised version). Technical Report 17-96, University of Kent at Canterbury, 1996.
10. E. Brinksma, J-P. Katoen, R. Langerak, and D. Latella. Performance analysis and true concurrency semantics. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-Time System Development*. World Scientific,, pages 309-337, 1994.

11. E. Brinksma, J-P. Katoen, R. Langerak, and D. Latella. A stochastic causality-based process algebra. *The Computer Journal*, 38:552–565, 1995.
12. R.T. Casley, R.F. Crew, J. Meseguer, and V.R. Pratt. Temporal structures. *Mathematical Structures in Computer Science*, 1:179–213, 1991.
13. P. Degano, R. De Nicola, and U. Montanari. On the consistency of 'truly concurrent' operational and denotational semantics (extended abstract). In *Third Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press , pages 133–141, 1988.
14. C.J. Fidge. A constraint-oriented real-time process calculus. In Diaz and Groz, editors, *Formal Description Techniques V*, volume C–10 of *IFIP Transactions*. North-Holland, pages 363–378, 1993.
15. E. Goubault. Durations for truly-concurrent transitions. In H.R. Nielson, editor, *Programming Languages and Systems — ESOP'96*, volume 1058 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 173-188, 1996.
16. J. Gunawardena. A dynamic approach to timed behaviour. In B. Jonsson and J. Parrow, editors, *Concur' 94: Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 178–193, 1994.
17. C.A.R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.
18. W. Janssen, M. Poel, Q. Wu, and J. Zwiers. Layering of real-time distributed processes. In H. Langmaack, W.-P. de Roever, and J. Vytopil, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 393–417, 1994.
19. A.S.A. Jeffrey, S. Schneider, and F.W. Vaandrager. A comparison of additivity axioms in timed transition systems. Technical Report CS-R9366, Centre for Mathematics and Computer Science, 1993.
20. J-P. Katoen. *Quantitative and Qualitative Extensions of Event Structures*. PhD thesis, University of Twente, CTIT Ph. D-thesis series No. 96-09, 1996.
21. J-P. Katoen, R. Langerak, and D. Latella. Modelling systems by probabilistic process algebra: An event structures approach. In R.L. Tenney, P.D. Amer, and M.Ü. Uyar, editors. *Formal Description Techniques VI*, volume C–22 of *IFIP Transactions*. North-Holland, pages 253-268, 1994.
22. J-P. Katoen, D. Latella, R. Langerak, and E. Brinksma. On specifying real-time systems in a causality-based setting. In B. Jonsson and J. Parrow, editors, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 1135 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 385–405, 1996.
23. R. Langerak. Bundle event structures: a non-interleaving semantics for LOTOS. In Diaz and Groz, editors, *Formal Description Techniques V*, volume C–10 of *IFIP Transactions*. North-Holland, pages 331–346, 1993.
24. R. Loogen and U. Goltz. Modelling nondeterministic concurrent processes with event structures. *Fundamenta Informaticae*, 14:39–74, 1991.
25. N.A. Lynch and F.W. Vaandrager. Action transducers and timed automata. In W.R. Cleaveland, editor, *Concur '92*, volume 630 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 436–455, 1992.
26. A. Maggiolo-Schettini and J. Winkowski. Towards an algebra for timed behaviours. *Theoretical Computer Science*, 103:335–363, 1992.
27. R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
28. F. Moller and C. Tofts. A temporal calculus of communicating systems. In Baeten and Klop, editors, *Concur '90: Theories of Concurrency – Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 401–415, 1990.
29. D.V.J. Murphy. Time and duration in noninterleaving concurrency. *Fundamenta Informaticae*, 19:403–416, 1993.
30. X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In de Bakker et al., editors, *Real-time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 526–548,1992.
31. M. Nielsen, G.D. Plotkin, and G. Winskel. Petri nets, event structures and domains, part 1. *Theoretical Computer Science*, 13:85–108, 1981.
32. G.M. Pinna and A. Poigné. On the nature of events: another perspective in concurrency. *Theoretical Computer Science*, 138:425–454, 1995.
33. G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
34. D.A. Schmidt. *Denotational Semantics: a methodology for language development*. Allyn and Bacon, 1986.
35. S. Schneider. An operational semantics for timed CSP. *Information and Computation*, 116:193–213, 1995.
36. C.A. Vissers. FDTs for open distributed systems, a retrospective and a prospective view. In L. Logrippo, R.L. Probert, and H. Ural, editors, *Protocol Specification, Testing and Verification X*. North-Holland, pages 341–362, 1990.

37. Y. Wang. Real-time behaviour of asynchronous agents. In Baeten and Klop, editors, *Concur '90: Theories of Concurrency – Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 502–520, 1990.
38. G. Winskel. An introduction to event structures. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*. Springer-Verlag, pages 364–397, 1989.
39. J.J. Žic. Time-constrained buffer specifications in CSP+T and timed CSP. *ACM Transactions on Programming Languages and Systems*, 16:1661–1674, 1994.