On Specifying Real-Time Systems in a Causality-Based Setting*

Joost-Pieter Katoen[a], Rom Langerak[a], Diego Latella[b] and Ed Brinksma[a]

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

[b] *CNUCE Istituto del CNR, Via Santa Maria 36, 56100 Pisa, Italy*

Abstract. Event structures are a prominent noninterleaving model for concurrency. Real-time event structures associate a set of time instants to events, modelling absolute time constraints, and to causal dependencies, modelling relative delays between causally dependent events. We introduce this novel temporal model and show how it can be used to provide a denotational semantics to a real-time variant of a process algebra akin to LOTOS. This formalism includes a timed-action prefix which constrains the occurrence time of actions, a timeout and watchdog (i.e., timed interrupt) operator. An event-based operational semantics for this formalism is presented that is shown to be consistent with the denotational semantics. As an example we use an infinite buffer with time constraints on the message latency and the rates of accepting and producing data.

1 Introduction

Timed extensions of interleaving models for concurrency have been investigated thoroughly in the last decade. Although there are many different ways in which time can be incorporated in labelled transition systems, the most prominent interleaving model, it seems that this issue is quite well-understood, cf. [2, 21].

The incorporation of quantitative information in noninterleaving models, such as event structures [27], pomsets [24], and Mazurkiewicz traces [19], has received scant attention in the literature. Since these models are attractive at the design stages in which the observational behaviour is no longer prevalent, but where the intensional system characteristics dominate, we argue that such models in particular should deal with issues like time and probability. In these design stages it is of utmost importance how actions are scheduled in time and with what probability certain alternative executions, which at a more high level of abstraction could be faithfully modelled by nondeterminism, can appear.

* 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.

This paper therefore proposes a real-time extension of (a variant of) event structures; probabilities are dealt with in [6, 15]. The real-time model is used as a vehicle to provide a denotational semantics to a temporal process algebra based on a kernel which is akin to LOTOS [4]. This formalism includes a timed action-prefix operator which constraints the occurrence time of actions, and a timeout and watchdog (i.e., timed interrupt) operator.

The inclusion of time in partial-order models is not new: e.g., extensions are known of pomsets [7], configurations [18], {and, or}-automata [12], sets of posets [13] and event structures [20]. In [16] we proposed a model of timed event structures with a notion of urgency. Such a notion is stronger than strictly necessary for modelling timeouts and makes urgent events have an unpleasant global impact. The timed extension of causal trees [10] resembles the model we present in this paper. We are, however, unaware of any proposal that incorporates time, timeouts, and watchdogs in a partial-order setting. These ingredients are considered to be essential to specify real-time systems.

We use *extended bundle event structures* [17], an adaptation of Winskel's event structures [27] to fit the specific requirements of multi-party synchronization and disruption ($[>$). Since we believe that both interleaving and noninterleaving models are legitimate and *complementary* in the system design process we also consider an event-based operational semantics for the real-time process algebra at hand which, by omitting event identifiers, results in an interleaving semantics. The two semantics are proven to coincide (i.e., strong timed (event) bisimulation equivalent) and thus can be used in a coherent way. This also facilitates the comparison of our timed partial order model and the wealth of existing timed interleaving models. (For space reasons we refer to the proofs of claims to [14].)

2 The language

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

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

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 $a \in \text{Act} \cup \{\tau\}$ 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 synchronization actions. $|||$ abbreviates $\|_\varnothing$, i.e., parallel composition without synchronization. $B[H]$ denotes the *relabelling* of B according to H where H : Act $\longrightarrow$ 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 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: ; , +, $\|$, [>, >>, $\backslash$ and []. Trailing 0s are usually omitted.

The standard (interleaving) semantics of PA is presented in Table 1, in the style of [23]. The special action δ indicates the *successful termination* action of a behaviour; we assume $\delta \notin \mathrm{Act}$. All relabelling functions H are extended to $\mathrm{Act} \cup \{\tau, \delta\}$ under the requirement that $H(\tau) = \tau, H(\delta) = \delta$ and for $a \in \mathrm{Act}$ we have $H(a) \notin \{\tau, \delta\}$. G^δ denotes $G \cup \{\delta\}$.

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

Table 1. Structured operational semantics of PA.

The real-time process algebra PA_R is obtained by generalising action-prefix and adding two timed operators $\rhd$ and $\blacktriangleright$ to PA. We use $\mathrm{Time} = \mathbb{R}^+ \cup \{0, \infty\}$ as time domain, T to range over $\mathcal{P}(\mathrm{Time})$, 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 $t \in T$. We write $(t)\ a$ for $([t, \infty))\ a$ and a for $(0)\ a$. $B_1 \overset{t}{\rhd} B_2$ denotes the *timeout* of B_1 by B_2 at time t; initially it behaves like B_1, but if B_1 does not perform any action before t (since the enabling of this behaviour) then the control is passed to B_2. At time t a nondeterministic choice between B_1 and B_2 appears. $\rhd$ is called a 'weak timeout' [21]. $\blacktriangleright$ is a *watchdog* operator; initially $B_1 \overset{t}{\blacktriangleright} B_2$ behaves like B_1 but at time t control is passed to B_2 provided B_1 is not yet successfully terminated. Note that in $B_1 \overset{t}{\rhd} B_2$ control is passed to B_2 only if B_1 does not perform any action—either internal or not—before t, whereas in $B_1 \overset{t}{\blacktriangleright} B_2$ control is passed to B_2 at time t, regardless of the activities of B_1 until time t (with the exception of termination).

An interaction can only occur when all participants are ready to engage in it. E.g., consider $a; (T_1)\ b \|_{\{a,b\}} a; (T_2)\ b$. If t_a denotes the time of occurrence of action a, action b is enabled at any time in $t_a + T_1 \cap t_a + T_2 = t_a + (T_1 \cap T_2)$, where $t + T$ denotes $\{t + t' \mid t' \in T\}$. Notice that interactions may become impossible

due to incompatible timing constraints in the participating behaviours. E.g., if $T_1 \cap T_2 = \varnothing$, action b can never occur. Before defining the denotational (Sect. 5) and operational (Sect. 8) semantics of PA_R, in the sequel we shortly recall extended bundle event structures and we introduce their real-time extension.

3 Extended bundle event structures

(Extended bundle) event structures [17] 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.

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[2].

Causality is represented by a binary relation, the *bundle relation*, denoted by $\mapsto$. For set X of events, that are pairwise in conflict, and event e, $X \mapsto e$ means 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. 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 *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 $\forall X \subseteq E, e \in E$ we have $X \mapsto e$ implies $\forall e', e'' \in X : e' \neq e'' \Rightarrow e' \rightsquigarrow e''$.

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 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, that is, $\overline{\sigma} \triangleq \{x_1, \ldots, x_n\}$. For non-empty

[2] The terminology 'asymmetric' is adopted from [17, 22].

sequence σ, let σ_i denote the prefix of σ up to the $(i-1)$-th element, that is, $\sigma_i \triangleq x_1 \ldots x_{i-1}$, 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 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 (i.e., $e_i \notin \overline{\sigma_i}$) and are conflict-free ($e_i \notin \mathsf{cfl}(\sigma_i)$). In addition, each event in the event trace is preceded in the sequence by a causal predecessor for each bundle pointing to it (i.e., $e_i \in \mathsf{sat}(\sigma_i)$).

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 1. Fig. 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. Fig. 1(b) has $\{e_a, e'_a\} \mapsto e_b$, $\{e_a\} \mapsto e_x$ and $\{e'_a\} \mapsto e_y$. Some event traces of Fig. 1(a) are $e_a\, e_b\, e_c$, $e_b\, e_d\, e_a$ and $e_b\, e_a$.

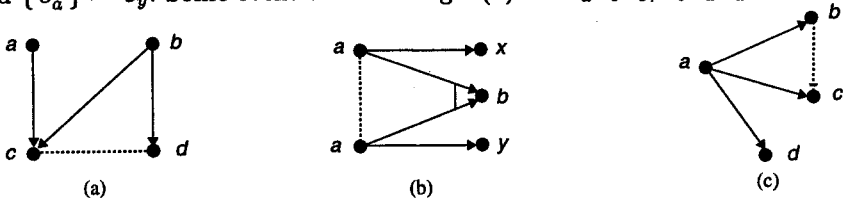

Fig. 1. Some example event structures.

Event structures can be used to provide a noninterleaving semantics to PA in a compositional way. For finite behaviours this is defined in Appendix A.
The expressions corresponding to Fig. 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 Real-time 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. Though it seems sufficient to only have time labels for initial events, synchronization of events makes it necessary to allow for equipping all events with time labels, including the non-initial ones.

We assume mappings $\mathcal{T}$ and $\mathcal{D}$ to associate a set of time instants, to bundles and events, respectively. A bundle $X \mapsto e$ with $\mathcal{T}((X, e)) = T$ is denoted by $X \overset{T}{\mapsto} e$; its interpretation is that if an event in X has happened at a certain time, then e

is enabled t time units later, for any $t \in T$. $\mathcal{D}(e) = T$ means that e can happen at any $t \in T$ from the beginning of the system, usually time 0.

In order to specify timeout mechanisms we use *urgent* events. These events are forced to occur once they are enabled.

Definition 3. A *real-time event structure* Γ is a quadruple $\langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ with $\mathcal{E}$, an event structure $(E, \rightsquigarrow, \mapsto, l)$, $\mathcal{D} : E \longrightarrow \mathcal{P}(\mathsf{Time})$, the *event delay* function, $\mathcal{T} : \mapsto \longrightarrow \mathcal{P}(\mathsf{Time})$, the *bundle delay* function, and $\mathcal{U} : E \longrightarrow \mathsf{Bool}$, the *urgency* predicate such that for all $e \in E$ with $\mathcal{U}(e)$:

1. $\forall e' \in E, X \subseteq E : ((e' \rightsquigarrow e \ \vee \ e \rightsquigarrow e') \ \wedge \ X \mapsto e) \ \Rightarrow \ (X \mapsto e' \ \vee \ X \rightsquigarrow e')$
2. $\exists t \in \mathsf{Time} : \mathcal{D}(e) \subseteq [t,t] \ \vee \ (\exists X \subseteq E : X \overset{T}{\mapsto} e \ \wedge \ T \subseteq [t,t])$.

Here, $X \rightsquigarrow e'$ equals $(\forall e'' \in X : e'' \rightsquigarrow e')$. Note that $\varnothing \rightsquigarrow e'$ for all e'.

The first constraint requires that the enablings of an urgent event e are either contained in the enablings of an event e' that it disables, i.e., $e' \rightsquigarrow e$, or that an enabling of e is disabled by e' (the case $e \rightsquigarrow e'$ is identical). This constraint enforces that as soon as e' is enabled either e is also enabled (provided e is not disabled in another way), or is permanently disabled, since some enabling of e is disabled (by e'). As a result the global impact of urgent events is limited; see also [16]. Thus, in order to decide whether e' can occur—once it is enabled—it suffices to consider the local (and urgent) disablings of e'.

The second constraint ensures that urgent events are enabled at a single time instant, if ever. The motivation for this constraint is that urgent events are used for the sole purpose of modelling timeouts, and a timeout typically can appear at a single time instant only.

Let EBES_R denote the class of real-time event structures. Bundle and event delays are depicted near to a bundle and event, respectively. Urgent events are denoted by open dots, other events by closed dots. Zero delays are omitted.

For events that have more than one bundle pointing to them we take the following interpretation. Consider $\{e_a\} \overset{T}{\mapsto} e_c$ and $\{e_b\} \overset{T'}{\mapsto} e_c$. If e_a happens at time t_a and e_b at time t_b, then e_c is enabled at any $t \in (t_a{+}T) \cap (t_b{+}T')$. When the intersection of two (or more) sets of time instants is empty this means that the event at hand cannot occur at any time and will be permanently disabled.

The notion of *timed event trace* is defined as a generalization of the notion of 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 $[\sigma] \triangleq e_1 \ldots e_n$. Let $\mathsf{time}(\sigma, e)$ denote the set of time instants at which $e \in \mathsf{en}([\sigma])$ could 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 intersecting $\mathcal{D}(e)$ with proper sets (H_1 and H_2 below) representing the constraints (ii) and (iii):

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

The notion of timed event trace is now defined as follows. Let $Min(T)$ denote the minimum of set T. For $T = \varnothing$, $Min(T) \triangleq \infty$.

Definition 4. A *timed event trace* of $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U}\rangle$ 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 $e_1 \ldots e_n \in T(\mathcal{E})$, $t_i \in \mathsf{time}(\sigma_i, e_i)$, for all $0 < i \leqslant n$, and

$$\forall i,e : (e \in \mathsf{en}([\sigma_i]) \ \wedge\ \mathcal{U}(e) \ \wedge\ (e_i \rightsquigarrow e \ \vee\ e \rightsquigarrow e_i)) \ \Rightarrow\ t_i \leqslant Min(\mathsf{time}(\sigma_i, e)).$$

Let $T_T(\Gamma)$ denote the set of timed event traces of Γ. The first two constraints are self-explanatory. The third constraint takes care of the fact that urgent events may prevent the events that they disable (or by which they are disabled) to occur after a certain time. That is, event e_i can occur at time t_i provided there is no enabled urgent event e that disables e_i (or that is disabled by e_i) and that (if it occurs) must occur before t_i.

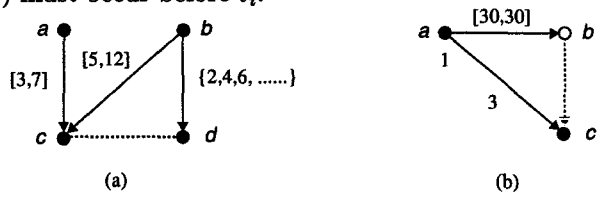

Fig. 2. Two real-time event structures.

Example 2. Fig. 2 depicts a real-time event structure with $\mathcal{T}((\{e_a\}, e_c)) = [3,7]$, $\mathcal{T}((\{e_b\}, e_c)) = [5,12]$ and $\mathcal{T}((\{e_b\}, e_d)) = \{2,4,6,\ldots\}$. Event delays are all zero. For the following sequences of timed events the conditions are given under which they are timed event traces of Fig. 2(a):

$$\begin{aligned}&(e_a,t_a)(e_b,t_b)(e_d,t_d) \ \text{ if } \ t_d \in \{t_b{+}2, t_b{+}4, \ldots\}, \text{ and}\\ &(e_a,t_a)(e_b,t_b)(e_c,t_c) \ \text{ if } \ max(t_a{+}3, t_b{+}5) \leqslant t_c \leqslant min(t_a{+}7, t_b{+}12).\end{aligned}$$

For Fig. 2(b) we obtain:

$$\begin{aligned}&(e_a,t_a)(e_c,t_c) \ \text{ if } \ t_a \geqslant 1 \ \wedge\ t_a{+}3 \leqslant t_c \leqslant t_a{+}30, \text{ and}\\ &(e_a,t_a)(e_b,t_b)(e_c,t_c) \ \text{ if } \ t_a \geqslant 1 \ \wedge\ t_b = t_a{+}30 \ \wedge\ t_c \geqslant max(t_a{+}3, t_b).\end{aligned}$$

Timed event traces do respect causality, but not necessarily time. That is, two (or more) independent events can occur in a trace in either order regardless of

their timing. For example, $(e_b,1)(e_a,3)$ and $(e_a,3)(e_b,1)$ are timed event traces of Fig. 2(a). The choices correspond to the possible interleavings of the causally independent events. Since the causal ordering between events implies their temporal ordering the causal ordering can never contradict the temporal order, see also [1].

The following result implies that for any ill-timed event trace σ there exists a corresponding time-consistent event trace σ', that can be obtained from σ by swapping repeatedly ill-timed pairs of timed events.

Theorem 5. $\forall t' < t : \sigma\ (e,t)(e',t')\ \sigma' \in T_T(\Gamma) \Rightarrow \sigma\ (e',t')(e,t)\ \sigma' \in T_T(\Gamma)$.

Note that the reverse implication does not hold; for instance, if e causally depends on e' then the order of events $e'\,e$ in a trace cannot be reversed since this would contradict their causal ordering.

5 Event structure semantics

This section presents a causality-based semantics for PA_R using real-time event structures. We define a mapping $\mathcal{E}_R[\![\ ]\!] : \mathsf{PA}_R \longrightarrow \mathsf{EBES}_R$. For convenience we use the denotational semantics $\mathcal{E}'[\![\ ]\!]$ for the untimed case which is defined in Appendix A. Recursion is dealt with in Sect. 6.

Definition 6. $\Phi : \mathsf{PA}_R \longrightarrow \mathsf{PA}$ is defined as follows:

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

$\Phi(B)$ is the untimed behaviour corresponding to B obtained by omitting all time annotations in B and converting $\rhd$ and $\blacktriangleright$ into $+$ and $[>$, respectively. The purpose of the internal event introduced by the timeout operator will be explained later on.

In the rest of this section let $\mathcal{E}_R[\![B_i]\!] = \Gamma_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$, 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 Appendix A and are used for real-time event structures in the same way, that is $init(\Gamma_i) \triangleq init(\mathcal{E}_i)$ and $exit(\Gamma_i) \triangleq exit(\mathcal{E}_i)$. Let E_U denote the (infinite) universe of events.

Definition 7. $\mathcal{E}_R[\![\ ]\!] : \mathsf{PA}_R \longrightarrow \mathsf{EBES}_R$ is defined for $\mathbf{0}$, $\surd$, and $(T)\ a$; as follows:

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

The semantics of $\mathbf{0}$ and $\surd$ is self-explanatory. In $\mathcal{E}_R[\![(T)\ a;\ B_1]\!]$ a bundle is introduced from a new event e_a (labelled a) to all events in Γ_1. The delay of each of these events, e, becomes relative to e_a, so each bundle $\{e_a\} \mapsto e$ is associated with a time delay $\mathcal{D}_1(e)$, and $\mathcal{D}(e)$ becomes Time. $\mathcal{D}(e_a)$ becomes T. In the untimed case it suffices to only introduce bundles from e to the initial events of Γ_1, cf. Appendix A. The bundles to all events of Γ_1 that are introduced in the timed case are used for the sole purpose of making delays relative to e_a. Fig. 3, e.g., shows (a) $\mathcal{E}_R[\![B]\!]$, and (b) $\mathcal{E}_R[\![([2,7))\ a;\ B]\!]$.

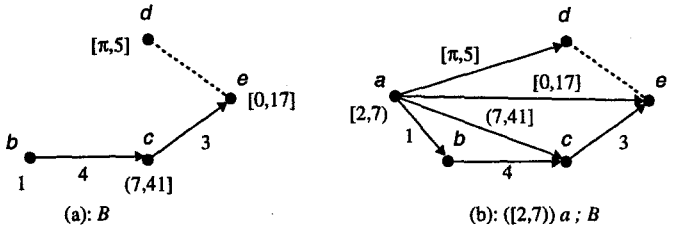

Fig. 3. Example of semantics for timed action prefix.

Definition 8. $\mathcal{E}_R[\![\ ]\!] : \mathsf{PA}_R \longrightarrow \mathsf{EBES}_R$ is defined for $\backslash$, $[]$, $+$, $>>$ and $[>$ as:

$$
\begin{aligned}
\mathcal{E}_R[\![B_1 \,\mathsf{op}\, B_2]\!] &\triangleq \langle \mathcal{E}'[\![\Phi(B_1 \,\mathsf{op}\, B_2)]\!], \mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle,\ \mathsf{op} \in \{+, [>\} \\
\mathcal{E}_R[\![\mathsf{op}\, B_1]\!] &\triangleq \langle \mathcal{E}'[\![\Phi(\mathsf{op}\, B_1)]\!], \mathcal{D}_1, \mathcal{T}_1, \mathcal{U}_1 \rangle \text{ for } \mathsf{op} \in \{\backslash, []\} \\
\mathcal{E}_R[\![B_1 >> B_2]\!] &\triangleq \langle (E_1 \cup E_2, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T}, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle \text{ where} \\
\rightsquigarrow &= \rightsquigarrow_1 \cup \rightsquigarrow_2 \cup \{(e, e') \mid e, e' \in \mathit{exit}(\Gamma_1) \wedge e \neq e'\} \\
\mapsto &= \mapsto_1 \cup \mapsto_2 \cup (\{\mathit{exit}(\Gamma_1)\} \times E_2) \\
l &= ((l_1 \cup l_2) \setminus (\mathit{exit}(\Gamma_1) \times \{\delta\})) \cup (\mathit{exit}(\Gamma_1) \times \{\tau\}) \\
\mathcal{D} &= \mathcal{D}_1 \cup (E_2 \times \{\mathsf{Time}\}) \\
\mathcal{T} &= \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{((\mathit{exit}(\Gamma_1), e), \mathcal{D}_2(e)) \mid e \in E_2\}.
\end{aligned}
$$

For op equal to choice or disrupt $\mathcal{E}_R[\![B_1 \,\mathsf{op}\, B_2]\!]$ 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 Γ_1 and Γ_2 are unaffected. Similarly, $\mathcal{E}_R[\![\ ]\!]$ is defined for relabelling and hiding. The events of $\mathcal{E}_R[\![B_1 >> B_2]\!]$ are those in $E_1 \cup E_2$. Bundles are introduced between the successful termination events of Γ_1 and the events in Γ_2. The reason for introducing bundles to all events of Γ_2 is to make the event delays in Γ_2 relative to the termination of Γ_1. This is similar as for timed action-prefix.

Now we consider parallel composition. Recall from Appendix A that events are pairs of events of Γ_1 and Γ_2, or with one component equal to $*$. The delay set of an event is the intersection of the delay sets of its components that are different from $*$. The time set associated with a bundle is equal to the intersection of the time sets 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 Γ_i.

For $E = (E_1 \cup \{*\}) \times (E_2 \cup \{*\}), (e_1, e_2) \in E$ and $X \subseteq E$ let for i=1, 2 projection be defined as $\mathsf{pr}_i((e_1, e_2)) \triangleq e_i$, if $e_i \neq *$ and $\mathsf{pr}_i(X) \triangleq \{\, \mathsf{pr}_i(e) \mid e \in X \cap \mathit{dom}(\mathsf{pr}_i) \,\}$. Let $\mathcal{T}_i((\varnothing, e_i)) = \mathsf{Time}$.

Definition 9. $\mathcal{E}_R[\![\ ]\!] : \mathsf{PA}_R \longrightarrow \mathsf{EBES}_R$ is defined for $||_G$ as follows:

$$
\begin{aligned}
\mathcal{E}_R[\![B_1 \,||_G\, B_2]\!] &\triangleq \langle \mathcal{E}'[\![\Phi(B_1 \,||_G\, B_2)]\!], \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle \text{ where} \\
\mathcal{D}((e_1, e_2)) &= \mathcal{D}_1(e_1) \cap \mathcal{D}_2(e_2) \text{ with } \mathcal{D}_i(*) = \mathsf{Time}. \\
\mathcal{T}((X, (e_1, e_2))) &= \mathcal{T}_1((pr_1(X), e_1)) \cap \mathcal{T}_2((pr_2(X), e_2)) \\
\mathcal{U}((e_1, e_2)) &= \mathcal{U}_1(e_1) \vee \mathcal{U}_2(e_2) \text{ with } \mathcal{U}_i(*) = \text{false}.
\end{aligned}
$$

Example 3. Consider $B_1 = ([1,7))\ a\,;\ (5)\ b \,||_b\, (\{1,3,6\})\ c\,;\ (7)\ b$ and $B_2 = ([4,9])\ a\,;\ (2)\ b \,||_b\, (((4,27])\ b + (3)\ d)$. Fig. 4 shows how $\mathcal{E}_R[\![B_1 \,||_{\{a,b\}}\, B_2]\!]$ is constructed from $\mathcal{E}_R[\![B_1]\!]$ and $\mathcal{E}_R[\![B_2]\!]$.

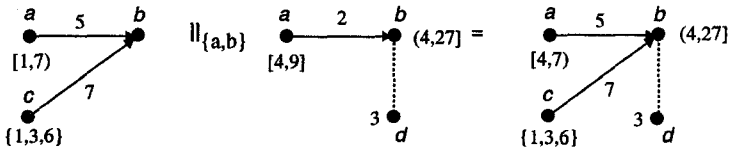

Fig. 4. Example of semantics for parallel composition.

In $\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!]$ a new internal, urgent event e_τ is introduced that models the expiration of the timer. Since either the timer expires or B_1 performs an initial action before (or at) t, event e_τ is put in mutual conflict with all initial events of Γ_1. The events of Γ_2 can only occur after the timeout; this is modelled in the same way as for action-prefix: a bundle $\{e_\tau\} \mapsto e$ is introduced for all $e \in \Gamma_2$. The delay of these bundles is determined as in the action-prefix case. The event delay of e_τ becomes $[t, t]$ such that it can only occur at t time units since the enabling of $\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!]$. So, $\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!]$ equals $\mathcal{E}_R[\![B_1 + ([t,t])\ \tau\,;\ B_2]\!]$ where τ is urgent.

Definition 10. $\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!] \triangleq \langle (E, \rightsquigarrow, \mapsto, l_1 \cup l_2 \cup \{(e_\tau, \tau)\}), \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ with

$E = E_1 \cup E_2 \cup \{e_\tau\}$ for some $e_\tau \in E_U \setminus (E_1 \cup E_2)$

$\rightsquigarrow = \rightsquigarrow_1 \cup \rightsquigarrow_2 \cup (init(\Gamma_1) \times \{e_\tau\}) \cup (\{e_\tau\} \times init(\Gamma_1))$

$\mapsto = \mapsto_1 \cup \mapsto_2 \cup (\{\{e_\tau\}\} \times E_2)$

$\mathcal{D} = \mathcal{D}_1 \cup \{(e_\tau, [t,t])\} \cup (E_2 \times \{\mathsf{Time}\})$

$\mathcal{T} = \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{((\{e_\tau\}, e), \mathcal{D}_2(e)) \mid e \in E_2\}$

$\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2 \cup \{(e_\tau, \mathrm{true})\}$.

Example 4. Let $B_1 = (2)\, a\,;\ (5)\, b \,|||\, ([6,21))\, c$ and $B_2 = (3)\, d\,;\ (2)\, g \,||_g\, ([27,41])\, g$. Fig. 5 illustrates how $\mathcal{E}_R[\![B_1 \overset{12}{\rhd} B_2]\!]$ is constructed from $\mathcal{E}_R[\![B_1]\!]$ and $\mathcal{E}_R[\![B_2]\!]$.

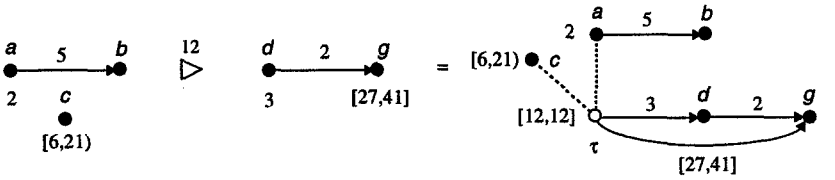

Fig. 5. Example of semantics for timeout opertor $\rhd$.

A similar approach could be taken for the watchdog operator (using $[>$ rather than $+$), but $B_1 \overset{t}{\blacktriangleright} B_2$ can also be modelled without urgent events. Consider $\mathcal{E}_R[\![B_1 [> B_2]\!]$, and (i) restrict all event delays in Γ_1 by $[0,t]$ ensuring that these events can only occur at time t at the latest, and (ii) postpone all events in Γ_2 by t such that these events can only occur from t on, cf. Fig. 6.

Definition 11. $\mathcal{E}_R[\![B_1 \overset{t}{\blacktriangleright} B_2]\!] \triangleq \mathcal{E}'[\![\Phi(B_1 \overset{t}{\blacktriangleright} B_2)]\!], \mathcal{D}, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle$ with

$\mathcal{D} = \{(e, \mathcal{D}_1(e) \cap [0,t]) \mid e \in E_1\} \cup \{(e, t + \mathcal{D}_2(e)) \mid e \in E_2\}$.

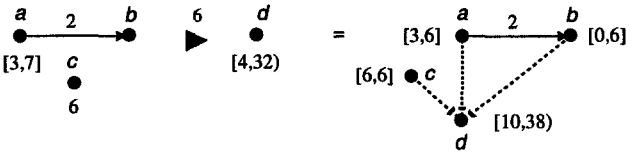

Fig. 6. Example of semantics for watchdog operator.

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.

$\mathcal{E}_R[\![P]\!]$ for $P := B$ is defined in the following way by using standard fixed point theory [26]. A complete partial order (c.p.o.) $\trianglelefteq$ is defined on real-time event structures with the empty event structure (i.e., $\mathcal{E}_R[\![\mathbf{0}]\!]$) as the least element $\perp$. Then for each definition $P := B$ a function $\mathcal{F}_B$ is defined that substitutes a real-time event structure for each occurrence of P in B, interpreting all operators in B as operators on real-time event structures. $\mathcal{F}_B$ is shown to be continuous, which means that $\mathcal{E}_R[\![P]\!]$ 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 continuous function $\mathcal{F}_B$. Further details can be found in [14].

Definition 12. Let $\Gamma_i = \langle (E_i, \rightsquigarrow_i, \mapsto_i, l_i), \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ for $i = 1, 2$. $\Gamma_1 \trianglelefteq \Gamma_2$ 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. $\forall e \in E_1 : \mathcal{T}_1((X \cap E_1, e)) = \mathcal{T}_2((X, e))$.

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 Γ_2 between events that are already in Γ_1. Similarly, the first constraint forbids the introduction of bundles in Γ_2 pointing to events in Γ_1 for which there exists no projected bundle in Γ_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 keep the same delay.

The l.u.b. $\bigsqcup_i \Gamma_i$ of a chain $\Gamma_1 \trianglelefteq \Gamma_2 \trianglelefteq \ldots$ can be characterized as follows. For the set of events, conflicts, labeling function, and event delays we simply take the union of all events, conflicts, labellings and event delays of the event structures in the chain. As bundles may grow this approach does not apply to the set of bundles. Suppose Γ_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 $\Gamma_1 \trianglelefteq \Gamma_2 \trianglelefteq \ldots$:

Definition 13. $\bigsqcup_i \Gamma_i \triangleq \langle (\bigcup_i E_i, \bigcup_i \rightsquigarrow_i, \mapsto, \bigcup_i l_i), \bigcup_i \mathcal{D}_i, \mathcal{T}, \bigcup_i \mathcal{U}_i \rangle$ with

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

Proposition 14. *$\bigsqcup_i \Gamma_i$ is the least upper bound of chain $\Gamma_1 \trianglelefteq \Gamma_2 \trianglelefteq \ldots$.*

Proposition 15. *For $\Gamma_1 \trianglelefteq \Gamma_2 \trianglelefteq \ldots$ a chain: $T_T(\bigsqcup_i \Gamma_i) = \bigcup_i \bigcap_{j \geqslant i} T_T(\Gamma_j)$.*

Definition 16. For $P := B$ a process definition let $\mathcal{E}_R[\![P]\!] \triangleq \bigsqcup_i \mathcal{F}_B^i(\perp)$.

7 Example: a time-constrained FIFO buffer

We show how PA_R and real-time event structures can be used to specify real-time systems by treating a time-constrained first-in first-out (FIFO) buffer. This example is taken from [28]; the only difference is that we consider a buffer of infinite length. A simple way to specify a FIFO buffer is by using an abstract data type *queue*:

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

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. $[b] \to E$ denotes that E can be executed if condition b holds.

The FIFO buffer should model a communication network with the following timing constraints [28]: (i) message latency in the range of 2 to 5 time units; (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:

$$\begin{aligned} TD &:= (wr_x \,;\, ([2,5])\ rd_x) \,|||\, TD \\ Wr &:= wr_x \,;\, Wr' \text{ where } Wr' := (1)\ wr_x \,;\, Wr' \\ Rd &:= rd_x \,;\, Rd' \text{ where } Rd' := (2)\ rd_x \,;\, Rd' \quad . \end{aligned}$$

The required buffer is obtained by putting these processes in parallel with *Fifo*: $\mathit{Fifo}(\langle\,\rangle) \,||\, Rd \,||\, Wr \,||\, TD$ where $||$ is a shorthand for $||_{\mathsf{Act}}$, i.e., full synchronization. This specification strongly resembles the timed CSP specification in [28].

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:

$$\begin{aligned} \mathit{Cell} &:= wp \,;\, \sum_{x \in D} wr_x \,;\, ((1)\ wn \,|||\, ([2,5])\ rp \,;\, rd_x \,;\, (2)\ rn) \\ \mathit{Chain} &:= (\mathit{Cell} \,||_{\{wn,rn\}}\, \mathit{Chain}[wp := wn, rp := rn]) \setminus \{wn, rn\} \\ \mathit{Buf} &:= \mathit{Chain} \setminus \{wp, rp\} \quad . \end{aligned}$$

The real-time event structures corresponding to the *Cell* and *Buf* processes are depicted in Fig. 7(a) and (b), respectively. The unlabelled, grey dots represent

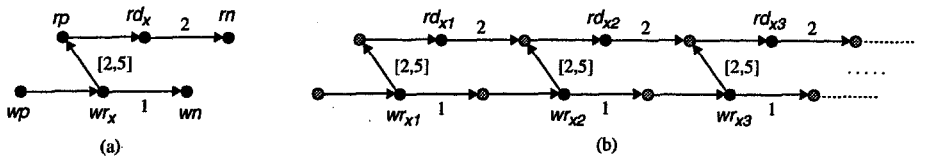

Fig. 7. Real-time event structure of a time-constrained FIFO buffer.

internal events. Process *Cell* describes a buffer cell allowing the writing and reading of a data value. The 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. Finally, process *Buf* hides the write-previous and read-previous actions of the front cell.

8 Event-based operational semantics

Most timed process algebras are based on an interleaving semantics. In order to facilitate a comparison with these existing approaches and to investigate the 'compatibility' of our proposal with the standard (interleaving) semantics of LOTOS we present an *event-based* operational semantics for PA_R. We define—as adopted from [5]—a transition system (in the sense of [23]) in which we keep track of the (times of) occurrence of actions rather than the actions themselves as is usual in structured operational semantics. This results in a *timed event transition system.*

Each occurrence of an action-prefix, $\surd$, and $\rhd$ is subscripted with an arbitrary but unique event identifier, denoted by a Greek letter. These identifiers play the rôle of event names. For $||_G$ new event names can be created. If $e \in B$ and $e' \in B'$, then possible new names for events in $B \,||_G\, B'$ are $(e, *)$ and $(*, e')$ for unsynchronized events and (e, e') for synchronized events. The operational semantics defines a set of transition relations $\xrightarrow{(e,a,t)}\!\!\!\!\rightarrow$. $B \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B'$ denotes that B can perform event e, labelled with action $a \in \mathsf{Act}^{\tau,\delta}$, at time $t \in \mathsf{Time}$, and subsequently evolve into B'. $\longrightarrow\!\!\!\!\rightarrow$ is the smallest relation closed under all inference rules of Table 2.

Let $\mathsf{ut}(B)$ denote the set of time instants at which B can *initially* perform an urgent event. Let PA_R^+ denote PA_R including the auxiliary operators ${}^t[\,]$ and ${}^t\{\,\}$ (see below).

Definition 17. $\mathsf{ut} : \mathsf{PA}_R^+ \longrightarrow \mathcal{P}(\mathsf{Time})$ is defined by:

$$
\begin{aligned}
\mathsf{ut}({}^t[B]) &\triangleq \{t'+t \mid t' \in \mathsf{ut}(B)\} \\
\mathsf{ut}(B_1 \;\mathsf{op}\; B_2) &\triangleq \mathsf{ut}(B_1) \cup \mathsf{ut}(B_2) \text{ for } \mathsf{op} \in \{+, [>, ||_G\} \\
\mathsf{ut}({}^t\{B\}) &\triangleq \{t' \in \mathsf{ut}(B) \mid t' \geqslant t\} \\
\mathsf{ut}(B_1 >> B_2) &\triangleq \mathsf{ut}(B_1) \\
\mathsf{ut}(\mathsf{op}\, B) &\triangleq \mathsf{ut}(B) \text{ for } \mathsf{op} \in \{\backslash, [\,]\}
\end{aligned}
$$

$$\mathsf{ut}(B_1 \overset{t}{\rhd} B_2) \triangleq \mathsf{ut}(B_1) \cup \{t\}$$
$$\mathsf{ut}(B_1 \overset{t}{\blacktriangleright} B_2) \triangleq \mathsf{ut}(B_1) \cup \mathsf{ut}({}^{t}[B_2])$$
$$\mathsf{ut}(P) \triangleq \mathsf{ut}(B) \text{ for } P := B.$$

For all other syntactical constructs let $\mathsf{ut}(B) \triangleq \varnothing$.

Let $\mathsf{mt}(B)$ abbreviate $\mathit{Min}(\mathsf{ut}(B))$, where Min of the empty set equals ∞. In order to let ut 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).

$$\vdash \surd_{\xi} \xrightarrow{(\xi,\delta,t)} \mathbf{0}$$
$$t \in T \vdash (T)\, a_{\xi}\,;B \xrightarrow{(\xi,a,t)} {}^{t}[B]$$
$$B \xrightarrow{(\xi,a,t)} B' \vdash {}^{t'}[B] \xrightarrow{(\xi,a,t+t')} {}^{t'}[B']$$
$$B_1 \xrightarrow{(\xi,a,t)} B_1' \quad t \leqslant \mathsf{mt}(B_2) \vdash B_1 + B_2 \xrightarrow{(\xi,a,t)} B_1'$$
$$B_2 \xrightarrow{(\xi,a,t)} B_2' \quad t \leqslant \mathsf{mt}(B_1) \vdash B_1 + B_2 \xrightarrow{(\xi,a,t)} B_2'$$
$$B_1 \xrightarrow{(\xi,a,t)} B_1' \quad a \neq \delta \vdash B_1 >> B_2 \xrightarrow{(\xi,a,t)} B_1' >> B_2$$
$$B_1 \xrightarrow{(\xi,\delta,t)} B_1' \vdash B_1 >> B_2 \xrightarrow{(\xi,\tau,t)} {}^{t}[B_2]$$
$$B_1 \xrightarrow{(\xi,a,t)} B_1' \quad (a \neq \delta \wedge t \leqslant \mathsf{mt}(B_2)) \vdash B_1 [> B_2 \xrightarrow{(\xi,a,t)} B_1' [> {}^{t}\{B_2\}$$
$$B_1 \xrightarrow{(\xi,\delta,t)} B_1' \quad t \leqslant \mathsf{mt}(B_2) \vdash B_1 [> B_2 \xrightarrow{(\xi,\delta,t)} B_1'$$
$$B_2 \xrightarrow{(\xi,a,t)} B_2' \quad t \leqslant \mathsf{mt}(B_1) \vdash B_1 [> B_2 \xrightarrow{(\xi,a,t)} B_2'$$
$$B \xrightarrow{(\xi,a,t)} B' \quad t \geqslant t' \vdash {}^{t'}\{B\} \xrightarrow{(\xi,a,t)} {}^{t'}\{B'\}$$
$$B_1 \xrightarrow{(\xi,a,t)} B_1' \quad a \notin G^{\delta} \vdash B_1 \|_G B_2 \xrightarrow{((\xi,*),a,t)} B_1' \|_G B_2$$
$$B_2 \xrightarrow{(\xi,a,t)} B_2' \quad a \notin G^{\delta} \vdash B_1 \|_G B_2 \xrightarrow{((*,\xi),a,t)} B_1 \|_G B_2'$$
$$B_1 \xrightarrow{(\xi,a,t)} B_1' \wedge B_2 \xrightarrow{(\psi,a,t)} B_2' \; a \in G^{\delta} \vdash B_1 \|_G B_2 \xrightarrow{((\xi,\psi),a,t)} B_1' \|_G B_2'$$
$$B \xrightarrow{(\xi,a,t)} B' \quad a \notin G \vdash B \setminus G \xrightarrow{(\xi,a,t)} B' \setminus G$$
$$B \xrightarrow{(\xi,a,t)} B' \quad a \in G \vdash B \setminus G \xrightarrow{(\xi,\tau,t)} B' \setminus G$$
$$B \xrightarrow{(\xi,a,t)} B' \vdash B[H] \xrightarrow{(\xi,H(a),t)} B'[H]$$
$$B_1 \xrightarrow{(\xi,a,t')} B_1' \quad t' \leqslant t \vdash B_1 \overset{t}{\rhd}_{\psi} B_2 \xrightarrow{(\xi,a,t')} B_1'$$
$$t \leqslant \mathsf{mt}(B_1) \vdash B_1 \overset{t}{\rhd}_{\psi} B_2 \xrightarrow{(\psi,\tau,t)} {}^{t}[B_2]$$
$$B_1 \xrightarrow{(\xi,a,t')} B_1' \quad (t' \leqslant t \wedge a \neq \delta) \vdash B_1 \overset{t}{\blacktriangleright} B_2 \xrightarrow{(\xi,a,t')} B_1' \overset{t}{\blacktriangleright} B_2$$
$$B_1 \xrightarrow{(\xi,\delta,t')} B_1' \quad t' \leqslant t \vdash B_1 \overset{t}{\blacktriangleright} B_2 \xrightarrow{(\xi,\delta,t')} B_1'$$
$$B_2 \xrightarrow{(\xi,a,t')} B_2' \quad t \leqslant \mathsf{mt}(B_1) \vdash B_1 \overset{t}{\blacktriangleright} B_2 \xrightarrow{(\xi,a,t+t')} {}^{t}[B_2']$$
$$B \xrightarrow{(\xi,a,t)} B' \quad (P := B) \vdash P_{\pi} \xrightarrow{(\pi\xi,a,t)} \pi(B')$$
$$B \xrightarrow{(\xi,a,t)} B' \vdash \pi(B) \xrightarrow{(\pi\xi,a,t)} \pi(B')$$

Table 2. Event-based operational semantics for PA_R.

$\surd$ can perform the successful termination action δ at any time t. $(T)\, a_{\xi}\,;B$ can perform event ξ at time t, $t \in T$, and evolves into ${}^{t}[B]$. ${}^{t'}[B]$ can be considered as behaviour B shifted t' time units in advance. That is, if B can perform event ξ, say, at time t, then ${}^{t'}[B]$ can perform ξ at time $t+t'$. Note that ${}^{t'}[B]$ is only

an auxiliary construct; it has no counterpart at the language level. The rules for parallel composition in which no synchronization takes place, for hiding, and for relabelling are straightforward extensions of the untimed rules. Synchronization can only take place when both participants can perform an equally labelled event whose label is in G (or equals δ) at time t. The rules for $>>$ are also a straightforward extension of the rules for the untimed case except that in case B_1 performs a successful termination action δ at time t, then $B_1 >> B_2$ evolves into ${}^t[\,B_2\,]$ rather than B_2. This represents that t time units have passed before B_2 can start with its execution.

The rules for $B_1 + B_2$ are somewhat adapted since (initial) urgent events in B_1 or B_2 can decide the choice. E.g., in (12) $a + ((18)\ b \overset{5}{\triangleright}_\chi ([1,7)\ c)$ event χ will occur at time 5, and resolve the choice in favour of B_2. In general, if B_1 performs an event at time t then $B_1 + B_2$ can perform the same provided that B_2 cannot perform an urgent event at any time earlier, i.e., if $t \leqslant \mathrm{mt}(B_2)$. By symmetry, a similar condition is obtained for B_2 performing an event. Similar conditions appear for $[>$, $\triangleright$, and $\blacktriangleright$.

If B_1 performs an event at t and evolves into B_1' then $B_1\,[>\,B_2$ can do the same while evolving into $B_1'\,[>\,{}^t\{\,B_2\,\}$. ${}^t\{\,B_2\,\}$ behaves like B_2 except that it is unable to perform events before t. The other inference rules for disrupt are straightforward extensions of the rules for the untimed case.

The inference rule for ${}^{t'}\{\,B\,\}$ is that if B can perform an event at time t, then ${}^{t'}\{\,B\,\}$ can do so if $t \geqslant t'$. Note that ${}^{t'}\{\,B\,\}$ is—like ${}^{t'}[\,B\,]$—an auxiliary operator that cannot be used by the specifier.

If B_1 performs an event at time t', with $t' \leqslant t$, and evolves into B_1' then $B_1 \overset{t}{\triangleright}_\psi B_2$ can do the same; in this case the possibility that B_2 happens is dropped since B_1 has performed an action before (or at) time t. At t the timeout event ψ can happen and the resulting behaviour is ${}^t[\,B_2\,]$, B_2 shifted t time units in advance. This can only be done if $t \leqslant \mathrm{mt}(B_1)$. This condition ensures that ψ is not performed if B_1 can perform an urgent event before t.

If B_1 performs an event (which is not a successful termination event) at time t', with $t' \leqslant t$, and evolves into B_1' then $B_1 \overset{t}{\blacktriangleright} B_2$ can do the same while evolving into $B_1' \overset{t}{\blacktriangleright} B_2$; the possibility for disruption (at time t) by B_2 remains. If B_1 terminates successfully at t', $t' \leqslant t$, disruption by B_2 becomes impossible (like for $B_1\,[>\,B_2$). If B_2 performs an event at t' and evolves into B_2' then $B_1 \overset{t}{\blacktriangleright} B_2$ can perform the same at $t + t'$ (provided B_1 cannot perform an urgent event before t) and evolves into ${}^t[\,B_2'\,]$.

It is assumed that each process instantiation of P is uniquely identified. Different occurrences of the same process instantiation should produce different event transitions. In addition, event transitions cannot be repeated. For $P :=$ $([2,7])\ a_\xi\,;\,P_\pi$ we first have an event transition with (ξ, a, t) for $t \in [2,7]$; 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. $\pi(B)$ is behaviour B where all event identifiers in B are prefixed with π.

Let $\mathsf{UE}(B)$ denote the set of urgent events in B. (This function can easily be defined by induction on the structure of B and is omitted here.) $\forall B \in \mathsf{PA}_R^+$:

Proposition 18. $(t \leqslant \mathsf{mt}(B)) \Leftrightarrow (\forall e \in \mathsf{UE}(B), t' < t : B \xrightarrow{(e,\tau,t')} \!\!\!\!/ \;\;\mapsto)$.

Let $\xrightarrow{\sigma}\!\!\mapsto$ denote the usual sequence-closure of the transition relation $\xrightarrow{(e,a,t)}\!\!\mapsto$. The consistency between the denotational and operational semantics of PA_R is

Theorem 19. $\forall B \in \mathsf{PA}_R : T_T(\mathcal{E}_R[\![B]\!]) = \{\sigma \mid \exists B' : B \xrightarrow{\sigma}\!\!\mapsto B'\}$

9 Conclusions and related work

This paper concerns a real-time extension of (a variant of) event structures, a partial-order model for concurrent systems. The original incentives of our work are to study the expressiveness of event structures to effectively support the specification of distributed systems and to facilitate formal representation of performance and reliability aspects. A secondary aim is to (formally) relate the real-time extension of event structures to interleaving models for concurrency such that partial-order and interleaving models can be used coherently in the system design process and can be compared in a perspicuous way.

To achieve this we proposed a real-time variant of extended bundle event structures, used this model for providing a (noninterleaving) denotational semantics to a real-time process algebraic formalism that includes a timeout and watchdog operator, and constructed a corresponding event-based operational semantics. This shows that event structures are suitable for modelling real-time systems. Both semantics are characterized by the absence of any mechanism that explicitly models the passage of time; time is treated as a parameter. The event-based operational semantics is a conservative extension of the standard interleaving operational semantics of LOTOS.

An interaction can take place if all participants can engage in it at the same time instant. The interaction cannot appear if such common time instant does not exist. Since in our model we do not have an explicit notion of the passage of time such an impossible interaction does not result in behaviours which do block the passage of time (*timelocks*) in the entire system—even in causally independent parts—but simply in the local impossibility to execute the event at hand.

The model based on timed-actions allows for the generation of ill-timed traces like in [1]. Recently, [11] proposed a timed process algebra with the theoretical CSP parallel operator that also includes ill-timed traces. In the proposals [1, 11] sub-processes have their independent local clock, and since local clocks are only synchronized at interaction, ill-timedness appears. We believe that the operational semantics presented in this paper is simpler by avoiding local clocks.

Ill-timedness is a phenomenon that is sometimes explicitly avoided by others (like in real-time ACP [3] and TIC [25]), since the precedence of timed events in the trace does not reflect the order in time. To our opinion ill-timed traces are not that obscure, since for each ill-timed trace there exists a corresponding time-consistent trace with the same timed events. Moreover, we think that the avoidance of them leads to a more complicated operational semantics.

Acknowledgements. Thanks to Pedro d'Argenio and Arend Rensink for suggestions.

A Denotational semantics of PA

In this appendix 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. 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.)

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

Definition 20. $\mathcal{E}[\![\,]\!] : \mathsf{PA} \longrightarrow \mathsf{EBES}$ is defined as follows:

$$
\begin{aligned}
\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 \text{ and}\\
\mathcal{E}[\![a\,;B_1]\!] &\triangleq (E_1 \cup \{e_a\},\rightsquigarrow_1,\mapsto,l_1 \cup \{(e_a,a)\}) \text{ for } e_a \in E_U \setminus E_1\\
\mapsto &= \mapsto_1 \cup (\{\{e_a\}\} \times \mathit{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 (\mathit{init}(\mathcal{E}_1) \times \mathit{init}(\mathcal{E}_2)) \cup (\mathit{init}(\mathcal{E}_2) \times \mathit{init}(\mathcal{E}_1))\\
\mathcal{E}[\![B_1 \setminus G]\!] &\triangleq (E_1,\rightsquigarrow_1,\mapsto_1,l) \text{ where}\\
&\quad (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 \mathit{exit}(\mathcal{E}_1) \wedge e \neq e'\}\\
\mapsto &= \mapsto_1 \cup \mapsto_2 \cup (\{\mathit{exit}(\mathcal{E}_1)\} \times \mathit{init}(\mathcal{E}_2))\\
l &= ((l_1 \cup l_2) \setminus (\mathit{exit}(\mathcal{E}_1) \times \{\delta\})) \cup (\mathit{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 \mathit{init}(\mathcal{E}_2)) \cup (\mathit{init}(\mathcal{E}_2) \times \mathit{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\\
&\quad \{(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\\
&\quad (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 \quad (\exists X_1 \subseteq E_1 : X_1 \mapsto_1 e_1 \wedge X = \{(e,e') \in E \mid e \in X_1\})\\
&\quad \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{aligned}
$$

Here we use a slight variant of $\mathcal{E}[\![\,]\!]$, denoted $\mathcal{E}'[\![\,]\!]$. $\mathcal{E}'[\![\,]\!]$ introduces not only bundles from e_a to the initial events of $\mathcal{E}_1$, but to all events in $\mathcal{E}_1$. Similarly, for $>\!>$ $\mathcal{E}'[\![\,]\!]$ introduces bundles from $\mathit{exit}(\mathcal{E}_1)$ to all events in $\mathcal{E}_2$. These additional bundles do not pose any problems, since $T(\mathcal{E}[\![B]\!]) = T(\mathcal{E}'[\![B]\!])$, for $B \in \mathsf{PA}$.

References

1. L. Aceto & D. Murphy. On the ill-timed but well-caused. In E. Best, ed, *Concur' 93*, LNCS 715: 97–111. Springer-Verlag, 1993.
2. R. Alur & D.L. Dill. A theory of timed automata. *Th. Comp. Sci.*, 126:183–235, 1994.
3. J.C.M. Baeten & J.A. Bergstra. Real time process algebra. *Formal Aspects of Computing*, 3(2):142–188, 1991.
4. T. Bolognesi & E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.
5. G. Boudol & I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. *Inf. & Comp.*, 114: 247–314, 1994.
6. E. Brinksma, J.-P. Katoen, R. Langerak & D. Latella. Performance analysis and true concurrency semantics. In T. Rus & C. Rattray, eds, *Theories and Experiences for Real-Time System Development*, pp. 309–337. World Scientific, 1994.

7. R.T. Casley, R.F. Crew, J. Meseguer & V.R. Pratt. Temporal structures. *Mathematical Structures in Computer Science*, 1(2):179–213, 1991.
8. J.W. de Bakker, W.-P. de Roever & G. Rozenberg, eds. *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, LNCS 354. Springer-Verlag, 1989.
9. M. Diaz & R. Groz, eds. *Formal Description Techniques V*. North-Holland, 1993.
10. C. Fidge. A constraint-oriented real-time process calculus. In [9], pp. 363–378.
11. R. Gorrieri, M. Roccetti & E. Stancampiano. A theory of processes with durational actions. *Th. Comp. Sci.*, 140:73–94, 1995.
12. J. Gunawardena. A dynamic approach to timed behaviour. In B. Jonsson & J. Parrow, eds, *Concur' 94: Concurrency Theory*, LNCS 836: 178–193. Springer-Verlag, 1994.
13. W. Janssen, M. Poel, Q. Wu & J. Zwiers. Layering of real-time distributed processes. In H. Langmaack, W.-P. de Roever & J. Vytopil, *Formal Techniques in Real-Time and Fault-Tolerant Systems*, LNCS 863: 393–417. Springer-Verlag, 1994.
14. J.-P. Katoen. *Quantitative and Qualitative Extensions of Event Structures*. PhD thesis, University of Twente, 1996.
15. J.-P. Katoen, R. Langerak & D. Latella. Modelling systems by probabilistic process algebra: An event structures approach. In R.L. Tenney, P.D. Amer & M.Ü. Uyar, eds, *Formal Description Techniques VI*, pp. 253–268. North-Holland, 1994.
16. J.-P. Katoen, D. Latella, R. Langerak, E. Brinksma & T. Bolognesi. A consistent causality-based view on a timed process algebra. In A. Cornell & D. Ionescu, eds, *Proc. 3rd Amast Workshop on Real-Time System Development*, pp. 212–227, 1996.
17. R. Langerak. Bundle event structures: a non-interleaving semantics for LOTOS. In [9], pp. 331–346.
18. A. Maggiolo-Schettini & J. Winkowski. Towards an algebra for timed behaviours. *Th. Comp. Sci.*, 103:335–363, 1992.
19. A. Mazurkiewicz. Basic notions of trace theory. In [8], pp. 285–363.
20. D.V.J. Murphy. Time and duration in noninterleaving concurrency. *Fund. Inf.*, 19:403–416, 1993.
21. X. Nicollin & J. Sifakis. An overview and synthesis on timed process algebras. In J.W. de Bakker et. al, eds, *Real-Time: Theory in Practice*, LNCS 600: 526–548. Springer-Verlag, 1992.
22. G.M. Pinna & A. Poigné. On the nature of events: another perspective in concurrency. *Th. Comp. Sci.*, 138(2):425–454, 1995.
23. G.D. Plotkin. A structural approach to operational semantics. Tech. Rep. DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
24. V.R. Pratt. Modeling concurrency with partial orders. *Int. J. of Parallel Programming*, 15(1):33–71, 1986.
25. J. Quemada, D. de Frutos & A. Azcorra. TIC: A TImed Calculus. *Formal Aspects of Computing*, 5:224–252, 1993.
26. D.A. Schmidt. *Denotational Semantics: a methodology for language development*. Allyn & Bacon, 1986.
27. G. Winskel. An introduction to event structures. In [8], pp. 364–397.
28. J.J. Žic. Time-constrained buffer specifications in CSP+T and timed CSP. *ACM Transactions on Programming Languages and Systems*, 16(6):1661–1674, 1994.