A True Concurrency Semantics for ET-LOTOS

Howard Bowman[a], Joost-Pieter Katoen[b]

[a] *University of Kent at Canterbury*
Canterbury, Kent CT2 7NF, United Kingdom
e-mail: H.Bowman@ukc.ac.uk

[b] *Lehrstuhl für Informatik VII, University of Erlangen-Nürnberg*
Martensstrasse 3, D-91058 Erlangen, Germany
e-mail: katoen@informatik.uni-erlangen.de

Abstract

One of the central objectives of the LOTOS restandardisation activity is to define an enhanced LOTOS language which supports real-time specification. The timed extension is based upon a timed LOTOS proposal ET-LOTOS. This report defines a (branching-time) non-interleaving semantics for ET-LOTOS without data. As a denotational model a suitable timed extension of Langerak's bundle event structures is used. For guarded recursive processes we show the consistency between our non-interleaving semantics and the ET-LOTOS interleaving semantics. Several examples illustrate the non-interleaving approach. Since our semantical model does not have an explicit notion of the passage of time (as opposed to the interleaving semantics) we are able to handle unguarded recursion and Zeno-behaviours in a perspicuous way.

Keywords: Consistency of semantics; Denotational semantics; Event structures; Failure semantics; Fixpoint theory; (enhanced) LOTOS; Operational semantics; Real-time; True concurrency

---

Contents

1 Introduction

The formal description technique LOTOS was standardised in 1987; for an introduction see [2]. However, it has been widely recognised that the language is deficient in a number of important respects. A particularly significant deficiency is the lack of support for real-time specification. This effectively prevents the description of a whole class of distributed systems applications, e.g. time dependent communication protocols such as those involving timeout behaviour. In addition, the flourishing of multimedia is giving added stimulus to the requirement to specify time-dependent aspects.

In response to this deficiency, one of the central objectives of the LOTOS restandardisation activity is to define an enhanced LOTOS language which supports real-time specification. The timed extension is based on a timed extension called ET-LOTOS.

In accordance with the standard "untimed" approach, the semantics for ET-LOTOS model concurrency as interleaving and are described in a structured operational style [26, 27]. In fact, this is by far the most common approach to giving semantics to timed process algebra [33]. However, although the interleaved operational approach has benefits, not least that it leads to simple and elegant semantic descriptions, it can be argued that at later stages of system development a true concurrency model is more appropriate.

Interleaved semantics reflect the *extensionalist* [30] view that formal specification should just describe the "observable behaviour" of systems. This inherently means that some aspects of causality are abstracted away from. For example, the following two behaviours would be identified:

$$a \;;\; b \;;\; \mathbf{stop} \;[]\; b \;;\; a \;;\; \mathbf{stop} \quad \text{and} \quad a \;;\; \mathbf{stop} \;|||\; b \;;\; \mathbf{stop}$$

even though the first contains two causalities, an instance of action a causing an instance of action b and an instance of action b causing an instance of action a, and the second contains no causalities. Abstracting, in this way, from the internal relationships between components

fits well into early phases of system development, such as the specification stage. Such phases typically involve the description of the global observable behaviour of systems and this is exactly what interleaved approaches model.

However, at later stages of system development explicit links to implementations have to be made. At this stage, semantic models must be capable of modelling the decomposition of systems into components which each have their own *local* state. In other words, they must accurately reflect the *distribution aspects* of the system under development. True concurrency models reflect such non-global interpretation; they are often categorised as *intensionalist* since they model the *internal* decomposition of systems. A further benefit of true concurrency models is that they limit the state space explosion problem; parallel composition generates the sum of the components states as opposed to the product, which is the case with interleaving semantics.

The motivation for true concurrency models carries over to the real-time setting. In fact it has been argued that the incorporation of real-time properties fits naturally with both the move to true concurrency models and with the focus on lower levels of system development [6, 14, 18]. In order to realise the benefits of the true concurrency approach in the timed LOTOS setting this paper presents a true concurrency semantics for ET-LOTOS.

A well accepted branching-time true concurrency model is *event structures*. These were initially developed by Nielsen, Plotkin & Winskel [32] as a link between Petri nets and Scott domain theory and have since then been extensively applied as a semantic model for process algebras, see for instance, [4, 28]. Within the event structures domain the *extended bundle event structures* developed by Langerak [23, 24] have particular relevance as they provide a semantic model tailored to the LOTOS disrupt and multi-way synchronisation operators. However, standard bundle event structures do not support real-time aspects.

In response, (several) real-time enhancements to bundle event structures have been developed [5, 8, 9, 21, 19]. However, although in these bodies of work the timed process calculus considered was strongly influenced by LOTOS, no one gives a semantics for ET-LOTOS. Furthermore, certain aspects of the ET-LOTOS model prompt a semantic treatment that is different to that included in either of the earlier models. Most significant amongst these aspects is the handling of urgency in combination with maximal progress. We will discuss this issue in Section 2.

In response to these observations, this paper makes the following contributions:

- We define a true concurrency semantics for ET-LOTOS based on a suitable timed extension of bundle event structures.
- For guarded recursive processes we relate our true concurrency semantics to the interleaving semantics of ET-LOTOS by Léonard & Leduc [26, 27] and thus prove consistency between the two approaches.

A limitation of our semantics is that we do not handle data and thus, we only give semantics to a "basic" subset of ET-LOTOS. This is unfortunate, but unavoidable, since a theory of event structures with data has not been devised and defining such a theory would be a significant undertaking in its own right. A consequence of this limitation is that we do not support the [*t in T*] notation. This powerful construct enables the relative time instant at which an action occurs to be recorded in the variable t and then referenced in the following behaviour.

Interestingly, some cases of degenerate behaviour that arise when ET-LOTOS is given an (operational or denotational) interleaving semantics do not arise in the true concurrency

setting. In our case the direct link between unguarded recursion and (global) time blocakge as present in the operational semantics [26, 27] is lost. In addition, instant recursive (taking no time between two successive recursive calls) and Zeno processes do not require a special treatment as opposed to the denottaional semantics based on timed failures [11, 12].

Although instant recursive, Zeno and unguarded processes are not realisable, it still makes sense to assign them an intuitive interpretation. In a specification style where process descriptions are viewed as constraints [36] these type of processes are of importance.

Structure of the paper. Section 2 introduces the basic ET-LOTOS language and illustrates its usefulness by means of an example from the field of distributed multimedia computing. Section 3 introduces the model of bundle event structures and presents a timed variant of this model that is tailored to the ET-LOTOS language. Section 4 defines the semantics of ET-LOTOS in terms of the timed variant of event structures of Section 3. The semantics of recursive processes is defined using the well-known fixed-point approach in the cpo-setting [35]. Several small examples illustrate the denotational semantics. Section 5 gives an event-based operational semantics for ET-LOTOS, which (for guarded recursive processes is proven in Section 6 to be consistent with the denotational semantics based on timed event structures. Section 7 compares our semantics with other ET-LOTOS semantics such as the interleaving semantics of LÉONARD & LEDUC [26, 27] and the timed failures semantics of BRYANS, DAVIES & SCHNEIDER [11, 12]. Finally, Section 8 concludes this report.

2 The TE-LOTOS Language

The timed calculus TE-LOTOS uses a number of sets of primitive entities. **Act** is a set of observable actions which does not contain the distinguished actions $\mathbf{i}$ and δ (these denote the internal action and successful termination respectively); $\mathbf{Act^i} = \mathbf{Act} \cup \{\mathbf{i}\}$; $\mathbf{Act}^\delta = \mathbf{Act} \cup \{\delta\}$; $\mathbf{Act}^{\mathbf{i},\delta} = \mathbf{Act} \cup \{\delta, \mathbf{i}\}$; D is a countable time domain; $D_\infty = D \setminus \{\infty\}$ and $D_{0\infty} = D \setminus \{0, \infty\}$.

The following abstract syntax defines the TE-LOTOS calculus that we consider, called *basic TE-LOTOS*. P ranges over specifications and Q ranges over behaviour expressions.

$$
\begin{array}{lcl}
P & ::= & Q \text{ where } \widetilde{x} := \widetilde{Q} \\
Q & ::= & \mathbf{stop} \mid \mathbf{exit}\ \{T\} \mid a\ \{T\}\ ;\ Q \mid \mathbf{Wait}(d)\ ;\ Q \mid Q\ []\ Q \mid \\
 & & Q\ |[\Gamma]|\ Q \mid \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q \mid Q >> Q \mid Q\ [> Q \mid Q[H] \mid x
\end{array}
$$

where $a \in \mathbf{Act^i}$; $\Gamma \subseteq \mathbf{Act}$, T ranges over time intervals $t^-..t^+$, $t^- \in D_\infty$, $t^+ \in D$, $d \in D_\infty$, $\widetilde{x}$ is a vector of process names, $\widetilde{Q}$ is a vector of behaviour expressions and $H : \mathbf{Act}^{\mathbf{i},\delta} \longrightarrow \mathbf{Act}^{\mathbf{i},\delta}$ such that $H(a) \notin \{\mathbf{i}, \delta\}$ for $a \in \mathbf{Act}$, and $H(a) = a$ for $a \in \{\mathbf{i}, \delta\}$.

Notice that in $\{t^-..t^+\}$ we do not prevent the situation, $t^+ < t^-$. This situation is interpreted as defining an empty interval. Conceptually, referencing an empty interval generates a deadlock; although, importantly, a time deadlock/blockage does not result. This point will be clarified shortly, when we discuss urgency. Thus, a behaviour such as:

$$a\ \{5..2\}\ ;\ Q$$

is behaviourally equivalent to **stop**. Such empty intervals play an important role in our semantic definitions.

Basic TE-LOTOS contains a deadlock behaviour **stop**; timed successful termination **exit** $\{T\}$; timed action prefix $a\ \{T\}\ ;\ Q$; a delay operator **Wait**$(d)\ ;\ Q$; choice $Q\ []\ Q$; parallel composition $Q\ |[\Gamma]|\ Q$; hiding **hide** Γ **in** Q; enabling $Q >> Q$; disabling $Q\ [>\ Q$ and process instantiation, x. For completeness we also consider relabelling $Q[H]$. For example, the normal LOTOS relabelling on process instantiation could be defined from $Q[H]$. Three of these operators have a time parameter:

- **exit** $\{t^-..t^+\}$ states that a δ will be offered at some time instant chosen from the interval $t^-..t^+$, i.e. between t^- and t^+ time units.
- $a\ \{t^-..t^+\}\ ;\ Q$ will offer the action a at some time instant in the interval $t^-..t^+$ and then, if the action is taken, the behaviour will evolve to Q.
- **Wait**$(d)\ ;\ Q$ will idle for d time units and then it will evolve to Q. This construct offers a single valued delay. In particular, basic TE-LOTOS does not support interval delays such as those considered in [8].

We assume the following abbreviations:

- $a\,\{t\}\ ;\ Q\ \equiv\ a\,\{t..\infty\}\ ;\ Q$
- $a\ ;\ Q\ \equiv\ a\,\{0\}\ ;\ Q\quad$ for $a \in \mathbf{Act}$
- $a\,(t)\ ;\ Q\ \equiv\ a\,\{t..t\}\ ;\ Q$
- $\mathbf{i}\ ;\ Q\ \equiv\ \mathbf{i}(0)\ ;\ Q$

$|[\varnothing]|$ is abbreviated by $|||$.

A crucial aspect of this language is the manner in which internal actions are handled; this is the much debated *urgency* issue. The handling of urgency is best illustrated by considering how three different forms of timed actions behave.

1. *Observable Actions* are offered to the environment, however the environment cannot be "forced" to take such actions. For example, the behaviour,

   $$a\{2..10\}\ ;\ b\ ;\ \mathbf{stop}$$

   will offer the action a to the environment between 2 and 10 time units from initiation of the behaviour. However, if the environment is not willing to perform an a action during this time period the behaviour will evolve to **stop**. Importantly, such an evolution does not prevent time from progressing; it creates a *local deadlock*, rather than a *global time blockage* (also called a *time lock*).

   This is in contrast to some of the alternative timed LOTOS proposals, such as [3], where time blockages can be created using urgent observable actions. By its nature a time blockage has a global effect. In particular, if a behaviour which blocks time is composed with a non-time blocking behaviour, in the composed behaviour time will not be able to progress beyond the time block. This occurs even if the non-blocking behaviour is not interacting with the time blocked behaviour.

   It should also be pointed out that δ is treated in the same non-urgent manner as observable actions. This is because successful terminations are synchronised through parallel composition in the usual way and thus, if they were urgent, could create time blockages.

2. *Explicit Internal Actions* can enforce urgency, however, they do not give pure maximal progress. By explicit internal actions we refer to behaviours in which internal actions are explicitly referenced, e.g.

$$\mathbf{i}\,\{\,2..10\,\}\,;\,b\,;\,\mathbf{stop}$$

This is in contrast to internal actions that arise through hiding, which we will call *hidden actions*. Firstly, since internal actions do not require any co-operation from the environment, enforcing urgency on such actions does not, in general, create time blocks. (This is not actually true since certain degenerate cases of internal behaviour can create time blocks; we will consider such cases in Section 7.) Thus in TE-LOTOS, internal actions must occur by the time the upper bound of their enabling is reached. The interpretation of the above example is that **i** may occur at any time between 2 and 10 time units from the initiation of the behaviour, but *must* have occurred once 10 time units have passed. Implicitly, the choice of time instant between 2 and 10 at which **i** occurs is non-deterministic. Thus, such intervals over internal actions can be used to specify non-deterministic timing intervals [8].

3. *Hidden Actions* also enforce urgency, but this time the urgency is stronger; hidden actions *must happen as soon as they become possible*. This interpretation corresponds to a pure maximal progress assumption [33]. As an illustration, consider the following behaviour:

$$\mathbf{hide}\ a\ \mathbf{in}\ (a\,\{\,2..10\,\}\,;\,b\,;\,\mathbf{stop})$$

This behaviour will perform an **i** action 2 time units after it is initiated and will then evolve to offering a b. However, importantly, it cannot perform an **i** action at any time after 2. A consequence of this approach is that the above behaviour is not equivalent to the behaviour highlighted in item 2.

In the sequel we will refer to urgency of explicit internal actions as *(internal) urgency* and urgency of hidden actions as *asap (as soon as possible)*.

Example 2.1. We illustrate the TE-LOTOS notation using an example from the field of distributed multimedia computing. The application of formal techniques to this area is being increasingly considered [7]. Distributed multimedia systems contain continuous flows of data with strict timing constraints, e.g. flows of audio or video. A multimedia stream is an abstraction of such a flow. We consider a very simple stream comprising a data source and a data sink communicating asynchronously over a medium. The channel is assumed to be infinite and may lose messages. Time units are milliseconds.

The top level behaviour of the stream is as follows,

$$\begin{aligned}&\mathbf{hide}\ sourceOut, sinkIn\ \mathbf{in}\ start\ ;\\ &\qquad (((sourceOut\,(0)\,;\,Source)\,|||\,Sink)\\ &\qquad |[sourceOut, sinkIn]|\\ &\qquad Channel)\end{aligned}$$

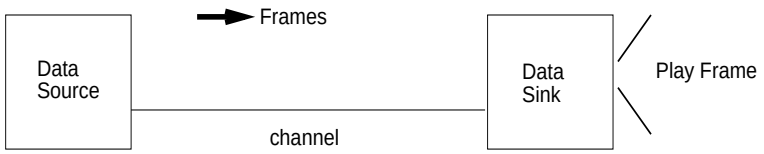


Figure 1: A multimedia stream

which composes the *Source*, *Sink* and *Channel* processes into a form equivalent to that depicted in Figure 1. The gates *sourceOut* and *sinkIn* are hidden from the external observer. The action *start* is offered until it is taken and the *sourceOut* action is offered immediately following *start* (as long as *Channel* is willing). The process *Source* has a very simple behaviour,

$$Source := sourceOut\,(50)\ ;\ Source$$

sourceOut actions are repeatedly offered at 50ms intervals. Notice that no error recovery is incorporated, i.e. acknowledgement or timeout schemes. This is typical of multimedia data, where temporal integrity is most important and retransmission is of limited value as such temporal integrity would be invalidated. The behaviour of the *Channel* can be specified as follows:

$$\begin{aligned} Channel := sourceOut\ ;\ ((&\mathbf{i}\,\{80..92\}\ ;\ sinkIn\,(0)\ ;\ \mathbf{stop} \\ &[]\,\mathbf{i}\,\{0..92\}\ ;\ \mathbf{stop}) \\ &|||\,Channel) \end{aligned}$$

So, the *Channel* will always offer a *sourceOut* and will then either internally evolve to offering a *sinkIn* or the *Channel* will internally decide to lose the message. The channel imposes a latency delay of between 80 and 92ms on each transmission. The action *sinkIn* will be offered, immediately after an internal action, at a time instant non-deterministically chosen from the interval $\{80..92\}$. The recursive call ensures that the channel is non-blocking. The data sink can be specified as,

$$\begin{aligned} Sink \quad := \quad & (sinkIn\,\{0..92\}\ ;\ (play\,(5)\ ;\ \mathbf{stop}\ |||\ Waiting)) \\ & []\,(\mathbf{i}\,(92)\ ;\ error\ ;\ \mathbf{stop}) \end{aligned}$$

$$\begin{aligned} Waiting \quad := \quad & (sinkIn\,\{0..50\}\ ;\ (play\,(5)\ ;\ \mathbf{stop}\ |||\ Waiting)) \\ & []\,(\mathbf{i}\,(50)\ ;\ error\ ;\ \mathbf{stop}) \end{aligned}$$

which either receives a *sinkIn* and *play*s the frame or returns an *error*. The process takes 5ms to process frames, i.e. the time between frames arriving and being played is 5ms. If either the first frame does not arrive within 92ms of the action start occuring or a frame does not arrive within 50ms of a previous frame the *Sink* will generate an *error*. *(End of example.)*

3 Time Extended Bundle Event Structures

3.1 Extended bundle event structures

Extended bundle event structures (which, for convenience, we simply call event structures in the rest of this paper) were developed as a non-interleaving model for LOTOS by LANGERAK [23, 24]. Since different occurrences of actions may have different causality relations to other actions, the central notion of event structures is that of an occurrence of an action, called *event*. For instance, in

$$a \;;\; b \;;\; \mathbf{stop} \,|||\, a \;;\; \mathbf{stop}$$

the occurrence of a on the left-hand side of $|||$ is needed to enable the occurrence of b, whereas the occurrence of a on the right-hand side is independent of b from a causality point of view. Each event is *labelled* by the action of which it models an occurrence. This labelling is not injective, since there may be several occurrences of the same action.

Event structures incorporate two kinds of relations between events: *conflict* and *bundle*. The *non-symmetric conflict* is a binary relation, denoted $\rightsquigarrow$, between events. 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 'non-symmetric'. $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$ is allowed and is equivalent to $e \# e'$, the more common symmetric conflict in other brands of event structures, like prime, flow and stable event structures[1].

Causality is represented by the *bundle relation*, denoted $\mapsto$, and relates a set of events to an event. 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). Set X is called the *bundle set*.

Definition 3.1 (Event structure)
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) *non-symmetric conflict* relation,
- $\mapsto \subseteq \mathcal{P}(E) \times E$, the *bundle* relation, and
- $l : E \longrightarrow \mathbf{Act}^{\mathbf{i},\delta}$, the *action-labelling* function

such that for all $X \subseteq E$ with $X \mapsto e$ we have $(X \times X) \setminus \mathrm{Id} \subseteq \rightsquigarrow$, where Id denotes the identity relation on E. □

The constraint specifies that for bundle $X \mapsto e$ all events in X are in mutual conflict. This constraint enables us to uniquely identify the causal predecessors of an event in a system run. (If this constraint is abandoned the interpretation of event structures becomes significantly more complicated, cf. the recent study of LANGERAK, BRINKSMA & KATOEN [25].) For the moment it suffices to consider finite event structures, i.e., event structures for which E is finite.

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

[1] The notion of non-symmetric conflict also appears in the context of event automata by PINNA & POIGNÉ [34].

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, if this does not introduce any confusion. ES denotes the class of event structures; $\mathcal{E}$ ranges over ES.

Example 3.2. An example (extended bundle) event structure is depicted in Figure 2. This event structure consists of the set of events $\{ e_a, e_b, e_{\mathbf{i}}, e_c \}$, conflicts $e_b \rightsquigarrow e_{\mathbf{i}}$ and $e_{\mathbf{i}} \rightsquigarrow e_b$, bundles $\{ e_a \} \mapsto e_b$ and $\{ e_b, e_{\mathbf{i}} \} \mapsto e_c$, and the obvious labelling. *(End of example.)*

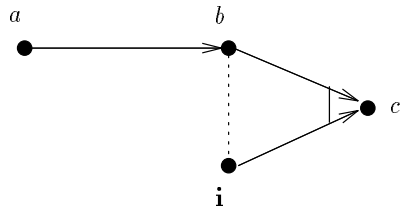


Figure 2: An example event structure

The interpretation of event structures is defined by means of the notion of *event trace*. An event trace captures in fact the concept of a system run and is a conflict-free sequence of events such that each event is preceded by its causal predecessors.

It is technically convenient to introduce the following abbreviations. 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 $0 < i \leqslant n+1$.

Definition 3.3 (Notations for event sequences)
For σ a sequence of events $e_1 \ldots e_n$ let

- $\mathsf{cfl}(\sigma) \triangleq \{ e \in E \mid \exists\, e_i \in \overline{\sigma} : e \rightsquigarrow e_i \}$
- $\mathsf{sat}(\sigma) \triangleq \{ e \in E \mid \forall\, X \subseteq E : X \mapsto e \;\Rightarrow\; X \cap \overline{\sigma} \neq \varnothing \}$, and
- $\mathsf{en}(\sigma) \triangleq \mathsf{sat}(\sigma) \setminus (\mathsf{cfl}(\sigma) \cup \overline{\sigma})$

□

$\mathsf{cfl}(\sigma)$ 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 (i.e, causal constraints) are 'satisfied'. $\mathsf{en}(\sigma)$ is the set of events enabled after σ.

Definition 3.4 (Event trace)
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 3.5. As an example, the event traces of Figure 2 are $e_a\, e_b\, e_c$, $e_a\, e_{\mathbf{i}}\, e_c$, $e_{\mathbf{i}}\, e_a\, e_c$, $e_{\mathbf{i}}\, e_c\, e_a$ and all prefixes of these event traces. Remark that e_c can only happen if either e_b or $e_{\mathbf{i}}$ has happened before. *(End of example.)*

It has been shown by Langerak [24, 23] that the kind of event structures introduced above can be used to provide a non-interleaving semantics to LOTOS in a compositional way.

He also showed that this alternative semantics is consistent with the standard interleaving semantics of LOTOS, which can be found in e.g. BOLOGNESI & BRINKSMA [2]. Other results on bundle event structures are e.g.: transformation laws preserving equivalence in terms of event traces (LANGERAK [24, 23]), various extensions with real-time (like BOWMAN [5], BOWMAN & DERRICK [8], BRINKSMA ET AL. [9], KATOEN ET AL. [21] and KATOEN ET AL. [19]), and a probabilistic extension (KATOEN, LANGERAK & LATELLA [20]). An interesting connection to performance analysis has been made: a stochastic extension of event structures has been defined where the delay of actions is determined by continuous random variables (BRINKSMA ET AL. [10]), and an algorithm has been developed for performing discrete-event simulation of (a subclass of) such event structures (KATOEN ET AL. [22]).

3.2 Adding time to event structures

Time is added to event structures in the following way. 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. The latter can be considered as absolute time constraints. [2]

In order to keep track of those events that must happen as soon as possible we use a set $\mathcal{I}$ (for immediate) of events. Since urgent events can easily be recognised by their label (they are all labelled by $\mathbf{i}$) there is no need to explicitly keep track of such a set.

Definition 3.6 (Time-extended event structure)
A time-extended event structure Ψ is a quadruple $\langle \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ with for $t^- \in D_\infty$ and $t^+ \in D$

- $\mathcal{E}$, an event structure $(E, \rightsquigarrow, \mapsto, l)$
- $\mathcal{A} : E \longrightarrow \{\, t^-..t^+ \,\}$, the *event-timing* function
- $\mathcal{R} : \mapsto \longrightarrow \{\, t^-..t^+ \,\}$, the *bundle-timing* function, and
- $\mathcal{I} \subseteq E$, the set of *immediate* events such that $\forall\, e \in \mathcal{I} : l(e) = \mathbf{i}$.

□

The constraint specifies that immediate events are internal. Let TES denote the class of time-extended event structures. Bundle $X \mapsto e$ with $\mathcal{R}((X, e))$ equal to T is denoted by $X \overset{T}{\mapsto} e$. Bundle and event delays are depicted near to a bundle and event, respectively, where timings equal to $\{\, 0..\infty \,\}$ are usually omitted and intervals $\{\, t..\infty \,\}$ are abbreviated as t. Intervals $\{\, t^-..t^+ \,\}$ are also denoted as $[t^-, t^+]$, or $[t^-, t^+)$ if $t^+ = \infty$. Immediate events are denoted by open dots, other events by closed dots. Let $t{+}T$ denote the set $\{\, t{+}t' \mid t' \in T \,\}$.

Example 3.7. An example time-extended event structure is depicted in Figure 3. This event structure is a timed extension of the plain event structure in Figure 2. More precisely, we have $\mathcal{A}(e_a) = \mathcal{A}(e_b) = \mathcal{A}(e_c) = \{\, 0..\infty \,\}$. $\mathcal{A}(e_{\mathbf{i}}) = \{\, 2..6 \,\}$. Events e_a, e_b, and e_c are non-immediate, so $e \notin \mathcal{I}$ for these events; $e_{\mathbf{i}} \in \mathcal{I}$. The bundle timings are: $\mathcal{R}((\{\, e_a \,\}, e_b)) = \{\, 2..7 \,\}$, $\mathcal{R}((\{\, e_b, e_{\mathbf{i}} \,\}, e_c)) = \{\, 12..\infty \,\}$. *(End of example.)*

The notion of *timed event trace* is defined as a generalisation of the notion of event trace. We first try to intuitively characterise the meaning of timed bundles and timed events. The

[2] An alternative way for having event delays is to explicitly model the start of the system by some fictitious event, see BOWMAN [5].

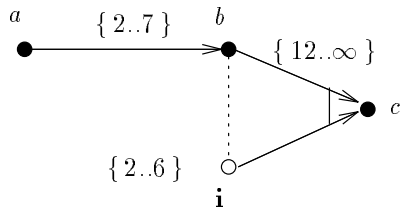


Figure 3: An example time-extended event structure

basic principle is that an event can happen once all its causal and timing constraints are fulfilled. For instance, $\mathcal{A}(e_c) = T$ means that e_c can happen at any $t_c \in T$ from the beginning of the system, which is assumed to be time 0. Suppose, in addition, we have $\{ e_a \} \overset{T'}{\mapsto} e_c$ and let e_a happen at time t_a. This bundle constrains the occurrence of e_c to any t_c for which $t_c \in t_a + T'$, corresponding to the idea that T' specifies the relative delay between e_a and e_c. Together with the previous timing constraint we get $t_c \in (t_a + T') \cap T$. Finally, add $e_b \rightsquigarrow e_c$ and assume that e_b and e_c both happen in a system run (thus e_c causally depends on e_b). Since we adopt the usual convention that causes should occur before their effects, $e_b \rightsquigarrow e_c$ puts the timing constraint $t_c \geqslant t_b$, or equally, $t_c \in [t_b, \infty)$. In total we now get $t_c \in (t_a + T') \cap T \cap [t_b, \infty)$. 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 because of incompatible timing constraints (as discussed earlier).

In order to facilitate the definition of timed event trace we introduce the following. Let (e, t) denote 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 event trace of σ, i.e., $[\sigma] \triangleq e_1 \ldots e_n$. Let $\mathcal{Z}(\sigma, e)$ 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. In accordance with the idea as sketched above, event e can occur if (i) its absolute delay $\mathcal{A}(e)$ 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 least at t_j. So, $\mathcal{Z}(\sigma, e)$ is obtained by intersecting $\mathcal{A}(e)$ with proper sets (H_1 and H_2 below) representing the constraints (ii) and (iii):

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

The notion of timed event trace is now defined as follows, where we take the convention that $\mathit{Min}(\varnothing) = \mathit{Max}(\varnothing) = \infty$.

Definition 3.8 (Timed event trace)
$\sigma = (e_1, t_1) \ldots (e_n, t_n)$ with $e_i \in E$, $t_i \in D_\infty$ is a timed event trace of $\Psi = \langle \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ iff for all $0 < i \leqslant n$:

1. $e_1 \ldots e_n$ is an event trace of $\mathcal{E}$
2. $\forall\, j : i < j \ \Rightarrow \ t_i \leqslant t_j$
3. $t_i \in \mathcal{Z}(\sigma_i, e_i)$
4. $\forall\, e \in \mathsf{en}([\sigma_i]) : l(e) = \mathbf{i} \ \Rightarrow \ t_i \leqslant f(\mathcal{Z}(\sigma_i, e))$

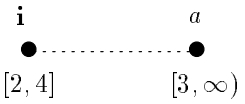
   where $f = \mathit{Min}$ if $e \in \mathcal{I}$, and Max otherwise.

□

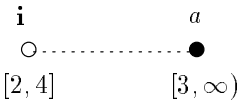
The first constraint requires a form of backwards compatibility with the untimed model: if all timings are deleted from σ, an event trace of $\mathcal{E}$ should be obtained. The second constraint requires time consistency and the third constraint says that an event can only happen at one of its possible event timings. The first three constraints do not take into account the fact that urgent events may prevent other events from occurring after a certain time. This is dealt with by the last constraint which says that trace σ_i may be extended with (e_i, t_i) iff there is no urgent (i.e., internal) event enabled after σ_i that should occur earlier than e_i. Since immediate events are forced to occur as soon as possible, while urgent events are forced to occur at the latest possible time, the last constraint distinguishes between these two cases. Notice that the last constraint together with the third constraint imply that immediate events should happen as soon as possible; they should not be delayed any further, that is:

$$e_i \in \mathcal{I} \;\Rightarrow\; t_i = Min\,(\mathcal{Z}(\sigma_i, e_i)).$$

Example 3.9. The (non-empty) timed event traces of the following structure



are $(e_{\mathbf{i}}, t)$ with $2 \leqslant t \leqslant 4$ and (e_a, t) with $3 \leqslant t \leqslant 4$. Notice that e.g. $(e_a, 5)$ is not a valid trace of this structure, since the internal event will be forced at time 4, thus disabling e_a. Now change the internal event into an immediate one, obtaining



This structure only has trace $(e_{\mathbf{i}}, 2)$, since the internal event is immediate (in addition to being urgent), so it is forced to occur at the first possible moment, $t = 2$, and there is no possibility for e_a to happen. *(End of example.)*

Example 3.10. The following example originates from Bolognesi, Lucidi & Trigila [3] and is known as the symmetric timeout example. It consists of two similar, interacting, processes where each process performs some local activity (modelled by actions a and b) with some unpredictable delay (e.g., delay $[0, \infty)$). Then they become ready to synchronise on a given action (modelled by event x) which should happen as soon as possible. This event x is hidden from the outside (so, internal). However, the processes are not willing to wait arbitrarily long on this interaction, and timeout after a certain delay (w_a and w_b, say). Figure 4 presents a time-extended event structure that models this situation: Suppose now that e_a happens at time t_a and e_b at time t_b. The timed event (x, t_x) belongs to a timed event trace iff

$$t_x \leqslant t_a + w_a \text{ and } t_x \leqslant t_b + w_b$$

since otherwise the left (or right, respectively) internal event will prevent x from happening. In addition, t_x should satisfy

$$t_x = Min\,(t_a + T_a \cap t_b + T_b)$$

which takes care of the immediateness of x and the time constraints with respect to x's causal predecessors. *(End of example.)*

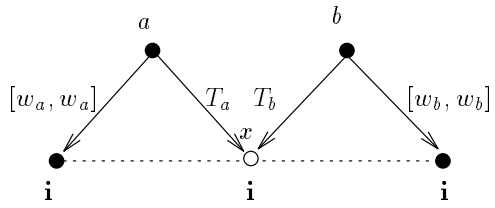


Figure 4: Symmetric timeout

4 Causal Semantics for ET-LOTOS

4.1 The semantics for finite behaviours

In this section we present a non-interleaving semantics for ET-LOTOS using time-extended event structures. We define a mapping $\mathcal{M}[\![\ ]\!] : \mathsf{ET-LOTOS} \longrightarrow \mathsf{TES}$. In this section we give the semantics for finite behaviours; recursive behaviours are dealt with in the next subsection. In the rest of this section let $\mathcal{M}[\![P]\!] = \Psi_P = \langle \mathcal{E}_P, \mathcal{A}_P, \mathcal{R}_P, \mathcal{I}_P \rangle$ with $\mathcal{E}_P = (E_p, \rightsquigarrow_P, \mapsto_P, l_P)$ and define $\mathcal{M}[\![Q]\!]$ analogously. For P and Q we assume $E_P \cap E_Q = \emptyset$; note that in case of name clashes this can easily be established by a suitable renaming. We use the following auxiliary definitions.

Definition 4.1 (Auxiliary definitions)
For $\Psi = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ let

- $\mathit{init}(\Psi) \triangleq \{ e \in E \mid \neg (\exists X \subseteq E : X \mapsto e) \}$
- $\mathit{exit}(\Psi) \triangleq \{ e \in E \mid l(e) = \delta \}$
- $\mathit{res}(\Psi) \triangleq \{ e \in E \mid \mathcal{A}(e) \neq D \}$

□

$\mathit{init}(\Psi)$ is the set of initial events of Ψ, $\mathit{exit}(\Psi)$ the set of successful termination events, and $\mathit{res}(\Psi)$ the events whose timing is restricted. We abbreviate $\mathit{init}(\Psi) \cup \mathit{res}(\Psi)$ by $\mathit{rin}(\Psi)$. Let E_U denote the universe of events.

Definition 4.2 (Inaction, termination and action-prefix)

$$
\begin{array}{rcl}
\mathcal{M}[\![\mathbf{stop}]\!] & \triangleq & \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle \\
\mathcal{M}[\![\mathbf{exit}\,\{ T \}]\!] & \triangleq & \langle \{ e_\delta \}, \emptyset, \emptyset, \{ (e_\delta, \delta) \}, \{ (e_\delta, \{ T \}) \}, \emptyset, \emptyset \rangle \\
 & & \text{where } e_\delta \in E_U \\
\mathcal{M}[\![a\,\{ T \}\ ;\ P]\!] & \triangleq & \langle E, \rightsquigarrow_P, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I}_P \rangle \text{ where} \\
E & = & E_P \cup \{ e_a \} \text{ for } e_a \in E_U \setminus E_P \\
\mapsto & = & \mapsto_P \ \cup\ (\{ \{ e_a \} \} \times \mathit{rin}(\Psi_P)) \\
l & = & l_P \cup \{ (e_a, a) \} \\
\mathcal{A} & = & \{ (e_a, \{ T \}) \} \cup (E_P \times D) \\
\mathcal{R} & = & \mathcal{R}_P \cup \{ ((\{ e_a \}, e), \mathcal{A}_P(e)) \mid e \in \mathit{rin}(\Psi_P) \}
\end{array}
$$

□

The empty event structure corresponds to **stop**. **exit** $\{ T \}$ results in a single non-immediate event, labelled δ with timing $\{ T \}$. For $a \{ T \} ; P$ a bundle is introduced from a new non-immediate event e_a (labelled a) to all initial events of Ψ_P (as e_a causally precedes them) and all events in Ψ_P that are time-restricted. For all these initial and time-restricted events e the delay is now relative to e_a, so each bundle $\{ e_a \} \mapsto e$ is associated with a time delay $\mathcal{A}_P(e)$, and $\mathcal{A}(e)$ becomes D. The timing of e_a becomes $\{ T \}$. In the untimed case it suffices to only introduce bundles from e to the initial events of Ψ_P, cf. LANGERAK [23, 24]. The bundles to all time-restricted events of Ψ_P that are introduced in the timed case are used for the sole purpose of making delays relative to e_a. Notice that the above construction applies to both observable and internal events.

Example 4.3. Figure 5(b) provides as an example the denotational semantics of $a \{ 2..3 \} ; P$ where it is assumed that the semantics of P is given as Figure 5(a). *(End of example.)*

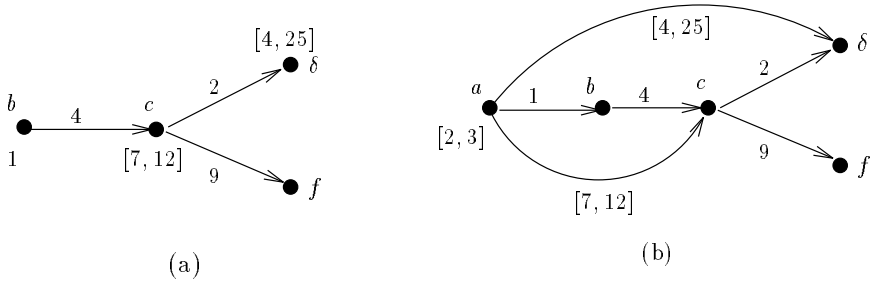


Figure 5: Example of semantics for action-prefix

Definition 4.4 (Delay, hiding and relabelling)

$$
\begin{array}{rcl}
\mathcal{M}[\![\mathbf{Wait}(d) ; P]\!] & \triangleq & \langle E_P, \rightsquigarrow_P, \mapsto_P, l_P, (+d) \circ \mathcal{A}_P, \mathcal{R}_P, \mathcal{I}_P \rangle \\
\mathcal{M}[\![\mathbf{hide}\ \Gamma\ \mathbf{in}\ P]\!] & \triangleq & \langle E_P, \rightsquigarrow_P, \mapsto_P, l, \mathcal{A}_P, \mathcal{R}_P, \mathcal{I} \rangle \text{ where} \\
 & & (l_P(e) \in \Gamma \ \Rightarrow \ l(e) = \mathbf{i}) \\
 & \wedge & (l_P(e) \notin \Gamma \ \Rightarrow \ l(e) = l_P(e)) \\
\mathcal{I} & = & \mathcal{I}_P \cup \{ e \in E_P \mid l_P(e) \in \Gamma \} \\
\mathcal{M}[\![P[H]]\!] & \triangleq & \langle E_P, \rightsquigarrow_P, \mapsto_P, H \circ l_P, \mathcal{A}_P, \mathcal{R}_P, \mathcal{I}_P \rangle
\end{array}
$$

□

The semantics of $\mathbf{Wait}(d) ; P$ is equal to $\mathcal{M}[\![P]\!]$ where all event delays are incremented by d ($\circ$ denotes usual function composition). Since bundle delays specify the relative delay between events, these timings are unaffected. $\mathcal{M}[\![\mathbf{hide}\ \Gamma\ \mathbf{in}\ P]\!]$ is identical to $\mathcal{M}[\![P]\!]$ except that events labelled with a label in Γ are now turned into internal, immediate events. $\mathcal{M}[\![P[H]]\!]$ is equal to $\mathcal{M}[\![P]\!]$ except that events are relabelled according to H.

Example 4.5. As an example of the denotational semantics consider some behaviour P, the semantics of which is depicted in Figure 6(a). After the hiding of action a the event structure of Figure 6(b) remains. *(End of example.)*

Definition 4.6 (Choice)

$$
\begin{array}{rcl}
\mathcal{M}[\![P [] Q]\!] & \triangleq & \langle E_P \cup E_Q, \rightsquigarrow, \mapsto_P \cup \mapsto_Q, l_P \cup l_Q, \mathcal{A}_P \cup \mathcal{A}_Q, \mathcal{R}_P \cup \mathcal{R}_Q, \mathcal{I}_P \cup \mathcal{I}_Q \rangle \\
\rightsquigarrow & = & \rightsquigarrow_P \cup \rightsquigarrow_Q \cup (\mathit{init}(\Psi_P) \times \mathit{init}(\Psi_Q)) \cup (\mathit{init}(\Psi_Q) \times \mathit{init}(\Psi_P))
\end{array}
$$

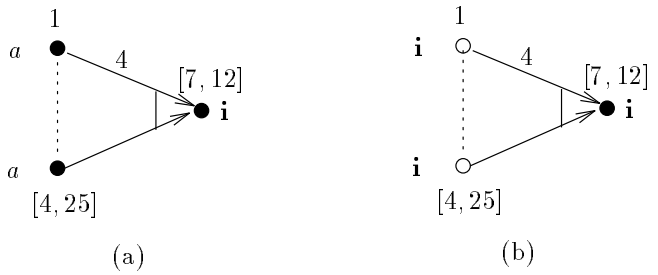


Figure 6: Example of semantics for hiding

□

$\mathcal{M}[\![P [] Q]\!]$ is equal to the component-wise union of $\mathcal{M}[\![P]\!]$ and $\mathcal{M}[\![Q]\!]$ extended with mutual conflicts between all initial events of $\mathcal{M}[\![P]\!]$ and $\mathcal{M}[\![Q]\!]$ such that in the resulting structure only one of P or Q can happen.

Definition 4.7 (Disrupt)

$$\mathcal{M}[\![P [> Q]\!] \triangleq \langle E_P \cup E_Q, \rightsquigarrow, \mapsto_P \cup \mapsto_Q, l_P \cup l_Q, \mathcal{A}_P \cup \mathcal{A}_Q, \mathcal{R}_P \cup \mathcal{R}_Q, \mathcal{I}_P \cup \mathcal{I}_Q \rangle$$
$$\rightsquigarrow \; = \; \rightsquigarrow_P \; \cup \; \rightsquigarrow_Q \; \cup (E_P \times \mathit{init}(\Psi_Q)) \cup (\mathit{init}(\Psi_Q) \times \mathit{exit}(\Psi_P))$$

□

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

Definition 4.8 (Enabling)

$$\begin{aligned}
\mathcal{M}[\![P >> Q]\!] &\triangleq \langle E_P \cup E_Q, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle \\
\rightsquigarrow &= \rightsquigarrow_P \; \cup \; \rightsquigarrow_Q \; \cup (\mathit{exit}(\Psi_P) \times \mathit{exit}(\Psi_P)) \setminus \mathrm{Id} \\
\mapsto &= \mapsto_P \; \cup \; \mapsto_Q \; \cup (\{ \mathit{exit}(\Psi_P) \} \times \mathit{rin}(\Psi_Q)) \\
l &= ((l_P \cup l_Q) \setminus (\mathit{exit}(\Psi_P) \times \{ \delta \})) \cup (\mathit{exit}(\Psi_P) \times \{ \mathbf{i} \}) \\
\mathcal{A} &= \mathcal{A}_P \cup (E_Q \times D) \\
\mathcal{R} &= \mathcal{R}_P \cup \mathcal{R}_Q \cup \{ ((\mathit{exit}(\Psi_P), e), \mathcal{A}_Q(e)) \mid e \in \mathit{rin}(\Psi_Q) \} \\
\mathcal{I} &= \mathcal{I}_P \cup \mathcal{I}_Q \cup \mathit{exit}(\Psi_P)
\end{aligned}$$

□

The events of $\mathcal{M}[\![P >> Q]\!]$ are those of $\mathcal{M}[\![P]\!]$ and $\mathcal{M}[\![Q]\!]$. Bundles are introduced from the successful termination events of $\mathcal{M}[\![P]\!]$ to the initial events and time-restricted events of $\mathcal{M}[\![Q]\!]$. (To create bundles, mutual conflicts are introduced between the successful termination events of $\mathcal{M}[\![P]\!]$.) The bundles to the initial events of $\mathcal{M}[\![Q]\!]$ model the fact that Q can start only if $\mathcal{M}[\![P]\!]$ has successfully terminated. Like for action-prefix the new bundles to the time-restricted events of $\mathcal{M}[\![Q]\!]$ are used for the sole purpose of making delays relative to the termination of $\mathcal{M}[\![P]\!]$. The timing of these bundles is treated similarly to the action-prefix case. The successful termination events of $\mathcal{M}[\![P]\!]$ are relabelled into internal events and become immediate.

Example 4.9. The denotational semantics of $P >> Q$ is illustrated in Figure 7. Notice that e_c is an initial event of $\mathcal{M}[\![Q]\!]$, whereas e_d is a time-restricted event of $\mathcal{M}[\![Q]\!]$. The extra

conflict between the two successful termination events of P is introduced to construct bundles as in the untimed event structure semantics for plain LOTOS.

(End of example.)

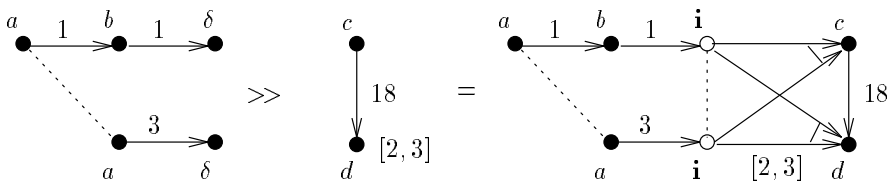


Figure 7: Example of semantics for enabling

The events of $\mathcal{M}[\![\, P \,|[\Gamma]|\, Q \,]\!]$ are constructed in the following way: an event e of $\mathcal{M}[\![\, P \,]\!]$ or $\mathcal{M}[\![\, Q \,]\!]$ that does not need to synchronise is paired with the auxiliary symbol $*$, and an event which is labelled with an action in Γ^δ (i.e., $\Gamma \cup \{\, \delta \,\}$) is paired with all events (if any) in the other process that are equally labelled. Thus events are pairs of events of $\mathcal{M}[\![\, P \,]\!]$ and $\mathcal{M}[\![\, Q \,]\!]$, 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 synchronise with the same event in the other process). A bundle is introduced such that if we take the projection on the component P (or Q) of all events in the bundle we obtain a bundle in $\mathcal{M}[\![\, P \,]\!]$ (or $\mathcal{M}[\![\, Q \,]\!]$, respectively). 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 P (or Q) of the events in the bundle, if this projection yields a bundle in $\mathcal{M}[\![\, P \,]\!]$ (or $\mathcal{M}[\![\, Q \,]\!]$, respectively).

Let $\mathcal{R}_i((\varnothing, e_i)) = D$. For $\Gamma \subseteq \mathbf{Act}$, $E_i^s \triangleq \{\, e \in E_i \mid l_i(e) \in \Gamma^\delta \,\}$ is the set of *synchronisation* events and $E_i^f \triangleq E_i \setminus E_i^s$ the set of *non-synchronising* events.

Definition 4.10 (Parallel composition)

$$
\begin{array}{rcl}
\mathcal{M}[\![\, P \,|[\Gamma]|\, Q \,]\!] & \triangleq & \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle \text{ where} \\
E & = & (E_P^f \times \{\, * \,\}) \cup (\{\, * \,\} \times E_Q^f) \cup \\
 & & \{\, (e_P, e_Q) \in E_P^s \times E_Q^s \mid l_P(e_P) = l_Q(e_Q) \,\} \\
(e_P, e_Q) \rightsquigarrow (e'_P, e'_Q) & \Leftrightarrow & (e_P \rightsquigarrow_P e'_P) \vee (e_Q \rightsquigarrow_Q e'_Q) \vee \\
 & & (e_P = e'_P \neq * \wedge e_Q \neq e'_Q) \vee (e_Q = e'_Q \neq * \wedge e_P \neq e'_P) \\
X \mapsto (e_P, e_Q) & \Leftrightarrow & (\exists\, X_P : X_P \mapsto_P e_P \wedge X = \{\, (e, e') \in E \mid e \in X_P \,\}) \\
 & & \vee\, (\exists\, X_Q : X_Q \mapsto_Q e_Q \wedge X = \{\, (e, e') \in E \mid e' \in X_Q \,\}) \\
l((e_P, e_Q)) & = & \mathsf{if}\ e_P = * \ \mathsf{then}\ l_Q(e_Q)\ \mathsf{else}\ l_P(e_P) \\
\mathcal{A}((e_P, e_Q)) & = & \mathcal{A}_P(e_P) \cap \mathcal{A}_Q(e_Q) \text{ with } \mathcal{A}_i(*) = D. \\
\mathcal{R}(X, (e_P, e_Q)) & = & \bigcap_{X_P \in S_P} \mathcal{R}_P(X_P, e_P) \cap \bigcap_{X_Q \in S_Q} \mathcal{R}_Q(X_Q, e_Q) \text{ with} \\
 & & S_P = \{\, X_P \subseteq E_P \mid X_P \mapsto_P e_P \ \wedge\ X = \{\, (e, e') \in E \mid e \in X_P \,\} \,\} \\
 & & S_Q = \{\, X_Q \subseteq E_Q \mid X_Q \mapsto_Q e_Q \ \wedge\ X = \{\, (e, e') \in E \mid e' \in X_Q \,\} \,\} \\
(e_P, e_Q) \in \mathcal{I} & \Leftrightarrow & e_P \in \mathcal{I}_P \ \vee\ e_Q \in \mathcal{I}_Q \text{ with } * \notin \mathcal{I}_i
\end{array}
$$

□

Remark that $(e_P, e_Q) \in \mathcal{I} \Rightarrow (e_P = * \vee e_Q = *)$ since immediate events are internal which cannot be the subject of synchronisation.

Example 4.11. The first example of parallel composition shows how the set of events is constructed when there is no synchronisation (i.e., $\Gamma = \varnothing$). In this case labels and timings of events and bundles remain unaffected. This is shown in Figure 8. *(End of example.)*

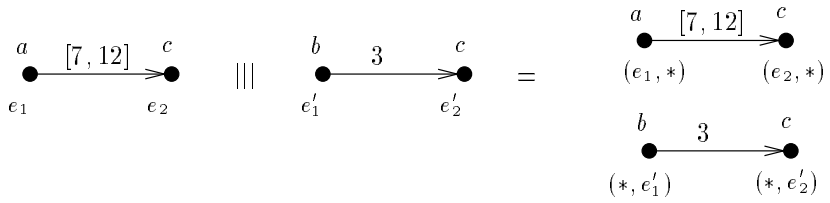


Figure 8: Example of semantics for parallel composition without synchronisation

Example 4.12. Synchronisation leads to pairing of events, and intersection of the event delays of its components, see Figure 9. *(End of example.)*

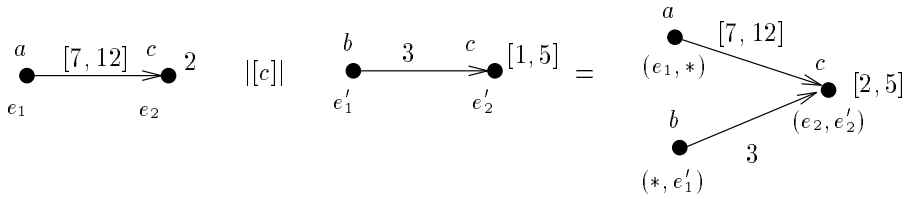


Figure 9: Example of semantics for parallel composition with synchronisation

Example 4.13. The next example (see Figure 10) shows how bundles can result from synchronisation. If one would resort to prime event structures [32] in this case, this would lead to the copying of event e'_2. *(End of example.)*

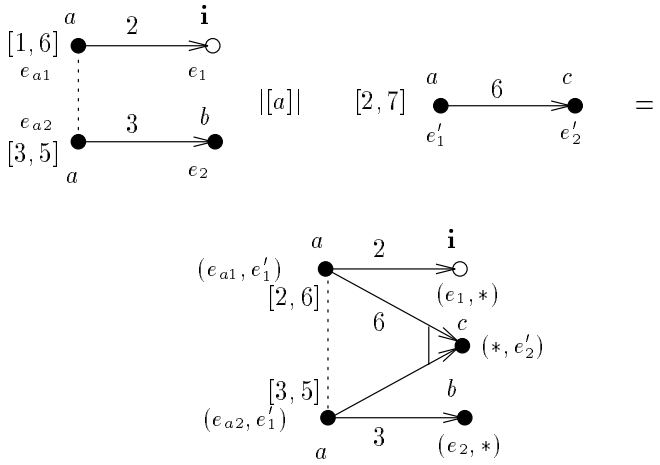


Figure 10: Example of semantics for parallel composition with synchronisation

The following lemma says that all events in a bundle set in a time-extended event structure originating from a ET-LOTOS expression are either immediate or not.

Lemma 4.14
For $\mathcal{M}[\![P]\!] = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ we have:

$$\forall\, X \subseteq E, e \in E : (X \mapsto e \ \wedge\ e', e'' \in X) \ \Rightarrow\ (e' \in \mathcal{I} \Leftrightarrow e'' \in \mathcal{I})$$

Proof:

It is quite straightforward to prove (by structural induction on P) that all events in a bundle set (i.e, X) are equally labelled. Since an event can only become immediate after hiding (either by the explicit hide operator or by enabling) this means that once some event in a bundle set is hidden, then all events in this bundle set become hidden, and thus immediate. *(End of proof.)*

4.2 The semantics of recursive processes

In this subsection we discuss the semantics of recursive processes. $\mathcal{M}[\![x]\!]$ for $x := Q$ is defined in the following way by using standard fixed point theory. A complete partial order (cpo) $\preceq$ is defined on time-extended event structures with the empty event structure (i.e., $\mathcal{M}[\![\mathbf{stop}]\!]$) as the least element $-$. Then for each definition $x := Q$ a function $\mathcal{F}_Q$ is defined that substitutes a time-extended event structure for each occurrence of x in Q, interpreting all operators in Q as operators on time-extended event structures. (Due to the compositionality of our semantics for finite processes this approach is possible.) When interpreted over time-extended event structures we overline the ET-LOTOS operator, e.g. $\overline{\mathbf{exit}\ \{T\}}$, $\overline{a\ \{T\}\ ;Q}$ and $Q\ \overline{|[\Gamma]|}\ Q$. $\mathcal{F}_Q$ is shown to be continuous w.r.t. $\preceq$, which means that $\mathcal{M}[\![x]\!]$ can be defined as the least upper bound (lub) of the chain (under $\preceq$) $-$, $\mathcal{F}_Q(-)$, $\mathcal{F}_Q(\mathcal{F}_Q(-))$, ... It is beyond the scope of this report to present the fixed point theory in its full details. However, the required semantic constructions are very closely related to those presented in Katoen [18] and have similarities to those given in Bowman [5].

We only consider direct recursion with a single process variable; that is, process definitions of the form: $x := ..x...x..$ Such an approach can be easily generalised to multiple process definitions and mutual recursion as shown by, amongst others, Manna, Ness & Vuillemin [29].

Notice that in order to deal with recursive behaviours we have to resort to event structures that can be infinite, i.e., they can have an infinite number of events and for bundles $X \mapsto e$ the bundle set X can be infinite.

We begin in standard fashion by defining an ordering on time-extended event structures, denoted $\preceq$.

Definition 4.15 (Ordering)
Let $\Psi_i = \langle E_i, \rightsquigarrow_i, \mapsto_i, l_i, \mathcal{A}_i, \mathcal{R}_i, \mathcal{I}_i \rangle$ for $i \in \{1, 2\}$. We define $\preceq$ as, $\Psi_1 \preceq \Psi_2$ if and only if,

1. $E_1 \subseteq E_2$
2. $\rightsquigarrow_1 = \rightsquigarrow_2 \cap (E_1 \times E_1)$
3. $\mapsto_1 = \{\, ((X \cap E_1), e) \mid e \in E_1 \ \wedge\ X \mapsto_2 e \,\}$
4. $l_1 = l_2 \restriction E_1$

5. $\mathcal{A}_1 = \mathcal{A}_2 \upharpoonright E_1$

6. $\forall e \in E_1 \ : \ \mathcal{R}_1((X \cap E_1, e)) = \bigcap\{\, \mathcal{R}_2((X, e)) \mid X \mapsto_2 e \,\}$

7. $\mathcal{I}_1 = \mathcal{I}_2 \cap E_1$

□

This ordering corresponds to an intuitive notion of when a time-extended event structure (TES) extends/enlarges another time-extended event structure. Implicitly, in $\Psi_1 \preceq \Psi_2$, Ψ_2 is a "larger" TES and Ψ_1 is a "smaller" TES. Such enlargement can be used to reflect the unfolding of a recursive process definition. Thus, the nth unfolding of a process definition will be larger, in the sense of $\preceq$, than the $n-1$ previous unfoldings. The definition imposes constraints on all the components of a time-extended event structure.

1. The events of Ψ_1 must be preserved in Ψ_2.

2. The conflicts of Ψ_1 must be preserved in Ψ_2 and between events that are inherited from Ψ_1, Ψ_2 cannot add any new conflicts.

3. All the bundles of Ψ_1 must be preserved in Ψ_2 with bundle sets, possibly, suitably enlarged and Ψ_2 cannot add new bundles to events inherited from Ψ_1.

4.,5.,7. Ensure that for all events inherited from Ψ_1, Ψ_2 preserves their labelling, timing and immediacy.

6. Ensure that for all bundles in Ψ_2 that are inherited from Ψ_1, the timing of these bundles is preserved.

Also in standard fashion, we introduce a least upper bound construction.

Definition 4.16 (Least upper bound)
For all $j \in \mathbb{N}$, $\Psi_j \in \mathsf{TES}$ and $\Psi_1 \preceq \Psi_2 \preceq \cdots$ a chain define
$\bigsqcup_i \Psi_i = \langle \bigcup_i E_i, \bigcup_i \rightsquigarrow_i, \mapsto, \bigcup_i l_i, \bigcup_i \mathcal{A}_i, \mathcal{R}, \bigcup_i \mathcal{I}_i \rangle$ where,

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

□

The Ψ_j's in this chain can be viewed as approximations to the meaning of a recursive process. Thus, each Ψ_j represents an unfolding of the recursive definition. This definition postulates that the TES $\bigsqcup_i \Psi_i$ is the least upper bound of such a chain, i.e. it is the smallest TES that is larger according to $\preceq$ than all TES's in the chain. The definition is again closely related to corresponding definitions in [18]. For events, conflict, labelling, event timing and immediacy, the definition is straightforward. In each of these cases any entity from the corresponding component of any TES in the chain is included (and nothing more). The definitions for $\mapsto$ and $\mathcal{R}$ are slightly more complicated. The former of these ensures that all bundles in the chain appear in $\bigsqcup_i \Psi_i$. Since bundles are preserved through $\preceq$ and bundle sets cannot decrease in size, we take the union of all the corresponding bundle sets of TESs in the chain. The definition of $\mathcal{R}$ ensures that the timing of bundles in $\bigsqcup_i \Psi_i$ is inherited from the timing of corresponding bundles in the chain.

We can now highlight the following results which justify these constructions.

Lemma 4.17

1. If $\Psi_1 \preceq \Psi_2 \preceq \ldots$ is a chain, then $\bigsqcup_i \Psi_i$ is the lub of the chain.

2. $\langle \mathsf{TES}, \preceq \rangle$ is a complete partial order with bottom element $- = \langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle$.

Proof:
Both these results are straightforward consequences of Lemmas to be found in [18] ((1) from Lemma 10.21 and (2) from Lemma 10.17). The only difference is that we have a seventh component to the semantic tuple, $\mathcal{I}$, but the results can be easily extended to accommodate this component. *(End of proof.)*

The following Lemma is also needed, it states that if Ψ_2 is larger than Ψ_1 and the events of Ψ_1 and Ψ_2 are the same then Ψ_1 and Ψ_2 are the same timed event structure.

Lemma 4.18
$(\Psi_1 \preceq \Psi_2 \ \wedge \ E_1 = E_2) \Longrightarrow \Psi_1 = \Psi_2$

Proof:

Once again, for all but the last component of Ψ_1 and Ψ_2 the proof of this result is the same as the corresponding Lemma in [18], Lemma 10.9. The result holds for the last component since $\mathcal{I}_1 = \mathcal{I}_2 \cap E_1 = \mathcal{I}_2 \cap E_2 = \mathcal{I}_2$. *(End of proof.)*

For the definition of $\mathcal{F}_Q$ we need the notion of renaming on time-extended event structures.

Definition 4.19 (Renaming of time-extended event structures)
For $\Psi = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ and occurrence identifier π let $\pi(\Psi) = \langle E', \rightsquigarrow', \mapsto', l', \mathcal{A}', \mathcal{R}', \mathcal{I}' \rangle$ with $E' = \{ \pi e \mid e \in E \}$, $\pi e \rightsquigarrow' \pi e'$ iff $e \rightsquigarrow e'$, $\pi X \mapsto' \pi e$ iff $X \mapsto e$, $l'(\pi e) = l(e)$, $\mathcal{A}'(\pi e) = \mathcal{A}(e)$, $\mathcal{R}'((\pi X, \pi e)) = \mathcal{R}((X, e))$, and $\pi e \in \mathcal{I}'$ iff $e \in \mathcal{I}$. □

Definition 4.20 (Function $\mathcal{F}_Q()$)
For process definition $x := Q$ and $\Psi \in \mathsf{TES}$ define $\mathcal{F}_Q(\Psi)$ by induction on the structure of Q as:

$$
\begin{aligned}
\mathcal{F}_{\mathbf{stop}}(\Psi) &\triangleq \overline{\mathbf{stop}} \\
\mathcal{F}_{\mathbf{exit}\{T\}}(\Psi) &\triangleq \overline{\mathbf{exit}\{T\}} \\
\mathcal{F}_{\mathsf{op}\, Q}(\Psi) &\triangleq \overline{\mathsf{op}}\, \mathcal{F}_Q(\Psi) \text{ for } \mathsf{op} \in \{ a\,\{T\};, \mathbf{Wait}(d)\ ;\ , \backslash \Gamma, [H] \} \\
\mathcal{F}_{Q\, \mathsf{op}\, R}(\Psi) &\triangleq \mathcal{F}_Q(\Psi)\, \overline{\mathsf{op}}\, \mathcal{F}_R(\Psi) \text{ for } \mathsf{op} \in \{ [\,], >>, [>, |[\Gamma]| \} \\
\mathcal{F}_{x_\pi}(\Psi) &\triangleq \pi(\Psi)
\end{aligned}
$$

□

$\mathcal{F}_Q(\Psi)$ for $x := Q$ replaces all process instantiations x_π in Q by an appropriately renamed copy of Ψ, $\pi(\Psi)$, while interpreting all ET-LOTOS operators in Q on time-extended event structures. The renaming of events ensures that all event identifiers in $\mathcal{F}_Q(\Psi)$ are unique. $\Psi = \mathcal{F}_Q(\Psi)$ has a unique fixed point if the function $\mathcal{F}_Q()$ is continuous w.r.t. $\preceq$. Similar to KATOEN [18] this boils down to prove that $\overline{\mathbf{exit}\ \{T\}}$, $\overline{a\ \{T\}\ ;}$, $\overline{|[\Gamma]|}$, etcetera, and $\pi()$ are continuous w.r.t. $\preceq$. In an approach inspired by WINSKEL [37], an alternative property is verified; this is called *continuity on events*.

Definition 4.21 (Continuity on events)
For $\langle \mathsf{TES}, \preceq \rangle$ a complete partial order and $F : \mathsf{TES} \longrightarrow \mathsf{TES}$, F is *continuous on events* if and

only if F is monotonic and for any chain $\Psi_1 \preceq \Psi_2 \preceq$ .... the following holds, $\underline{E}(F(\bigsqcup_i \Psi_i)) \subseteq \underline{E}(\bigsqcup_i F(\Psi_i))$, where $\underline{E}(\langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I}\rangle) = E$. □

The following lemma ensures that proving continuity on events is sufficient.

Lemma 4.22

For $\langle \mathsf{TES}, \preceq\rangle$ a complete partial order and $F : \mathsf{TES} \longrightarrow \mathsf{TES}$, F is continuous if and only if F is continuous on events.

Proof:

The proof of this result uses Lemma 4.18 and is similar to the proof of Lemma 10.11 of [18]. *(End of proof.)*

In order to complete the fixed point theory we have to prove that all operators of ET-LOTOS, when interpreted as operators over time-extended event structures, are indeed continuous over events. This will ensure that $\mathcal{F}_Q(\Psi)$ is continuous. Theorem 4.24 will verify this, but first a simple lemma.

Lemma 4.23

For $i \in \{1,2\}$ let $\Psi_i = \langle E_i, \rightsquigarrow_i, \mapsto_i, l_i, \mathcal{A}_i, \mathcal{R}_i, \mathcal{I}_i\rangle$; then $\Psi_1 \preceq \Psi_2$ implies $exit(\Psi_2) \cap E_1 = exit(\Psi_1)$.

Proof:

$$
\begin{aligned}
& exit(\Psi_2) \cap E_1 \\
= \quad & \{ \text{ definition of } exit \ \} \\
& \{e \mid e \in E_2 \ \wedge \ l_2(e) = \delta \ \wedge \ e \in E_1\} \\
= \quad & \{ \ E_1 \subseteq E_2; l_1 = l_2 \restriction E_1 \ \} \\
& \{e \mid e \in E_1 \ \wedge \ l_1(e) = \delta\} \\
= \quad & \{ \text{ definition of } exit \ \} \\
& exit(\Psi_1)
\end{aligned}
$$

(End of proof.)

Theorem 4.24 (Continuity of ET-LOTOS operators)

The operators of ET-LOTOS, i.e. $\overline{\textbf{stop}}$, $\overline{\textbf{exit}\ \{T\}}$, $\overline{a\ \{T\}\ ;}$, $\overline{\textbf{Wait}(d)\ ;}$, $\overline{[]}$, $\overline{|[\Gamma]|}$, $\overline{\textbf{hide}\ \Gamma\ \textbf{in}}$, $\overline{>>}$, $\overline{[>}$, and the renaming operator $\pi()$ are continuous w.r.t. the complete partial order $\langle \mathsf{TES}, \preceq\rangle$.

Proof:

Once again we can largely reuse the proofs to be found in [18] (Chapter 10). Compared to the definitions in [18], apart from minor changes (that do not affect continuity) to the semantics of action prefix and enabling, the only operators that have new semantic definitions here are exit, delay, hiding and enabling. We give the proofs for delay and enabling; the proofs for exit and hiding are both straightforward.

Using Lemma 4.22 it suffices to prove these operators are continuous on events. In the following, for $i \in \{1,2\}$, $\Psi_i = \langle E_i, \rightsquigarrow_i, \mapsto_i, l_i, \mathcal{A}_i, \mathcal{R}_i, \mathcal{I}_i\rangle$, $\Psi'_i = \langle E'_i, \rightsquigarrow'_i, \mapsto'_i, l'_i, \mathcal{A}'_i, \mathcal{R}'_i, \mathcal{I}'_i\rangle$ and $\Psi = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I}\rangle$.

1. Delay. We first show that $\overline{\textbf{Wait}(d)\ ;}\Psi$ is monotonic. So, let $\Psi_1 \preceq \Psi_2$, $\Psi'_1 = \overline{\textbf{Wait}(d)\ ;}\Psi_1$ and $\Psi'_2 = \overline{\textbf{Wait}(d)\ ;}\Psi_2$. Now the semantics for the delay operator leave all elements of the semantic tuple unchanged apart from $\mathcal{A}$. Thus, this is the only component we need to consider, but by observing that new events are not added by the delay operator, we can derive the following:

$$\mathcal{A}'_2 \restriction E'_1 = \mathcal{A}'_2 \restriction E_1 = ((+d) \circ \mathcal{A}_2) \restriction E_1 = (+d) \circ (\mathcal{A}_2 \restriction E_1) = (+d) \circ \mathcal{A}_1 = \mathcal{A}'_1$$

which justifies monotonicity. In addition, since the events component of the semantic tuple is not changed when applying the delay operator, it can easily be shown that $\underline{E}(\overline{\mathbf{Wait}(d)\ ;\ \bigsqcup_i \Psi_i}) = \underline{E}(\bigsqcup_i \overline{\mathbf{Wait}(d)\ ;\ \Psi_i})$. Thus, the delay operator is continuous on events.

2. Enabling. We first show that $\overline{>>}$ is monotonic in its left argument. So, let $\Psi_1 \preceq \Psi_2$, $\Psi_1' = \Psi_1 \overline{>>} \Psi$ and $\Psi_2' = \Psi_2 \overline{>>} \Psi$. Now the semantics of enabling are the same as those of [18] in all but the last component, the immediacy component. Thus, we only consider this here.

$$
\begin{array}{ll}
 & \mathcal{I}_2' \cap E_1' \\
= & \quad \{ \text{ semantics of } >> \} \\
 & (\mathcal{I}_2 \cup \mathit{exit}(\Psi_2) \cup \mathcal{I}) \cap (E_1 \cup E) \\
= & \quad \{ \text{ set calculus } \} \\
 & (\mathcal{I}_2 \cap (E_1 \cup E)) \cup (\mathit{exit}(\Psi_2) \cap (E_1 \cup E)) \cup (\mathcal{I} \cap (E_1 \cup E)) \\
= & \quad \{ \text{ the events of } \Psi_2 \text{ and } \Psi \text{ are disjoint; likewise for } \Psi_1 \text{ and } \Psi \ \} \\
 & (\mathcal{I}_2 \cap E_1) \cup (\mathit{exit}(\Psi_2) \cap E_1) \cup \mathcal{I} \\
= & \quad \{ \text{ definition of } \preceq \text{; Lemma 4.23 } \} \\
 & \mathcal{I}_1 \cup \mathit{exit}(\Psi_1) \cup \mathcal{I} \\
= & \quad \{ \text{ semantics of } >> \} \\
 & \mathcal{I}_1'
\end{array}
$$

which justifies monotonicity in the left argument. Continuity on events in the left argument follows since,

$$\underline{E}(\textstyle\bigsqcup_i \Psi_i \overline{>>} \Psi) = (\bigcup_i E_i) \cup E = \bigcup_i (E_i \cup E) = \underline{E}(\bigsqcup_i (\Psi_i \overline{>>} \Psi))$$

Monotonicity and continuity of events in the right argument of $>>$ is straightforward.

(End of proof.)

Using Theorem 4.24 we can now define, in the context of a process definition $x := Q$, the meaning of process instantiation as:

Definition 4.25 (Process instantiation)
$\mathcal{M}[\![\, x \,]\!] \triangleq \bigsqcup_i \mathcal{F}_Q^i(-)$ □

From the theory we have set up this definition characterises the (unique) least time-extended event structure, according to $\preceq$, that satisfies $\Psi = \mathcal{F}_Q(\Psi)$.

Example 4.26. We illustrate this definition with a simple example of a recursive process:

$$x := a\,\{2..4\}\ ;\ (\mathbf{hide}\ a\ \mathbf{in}\ (c\,\{3..6\}\ ;\ \mathbf{stop}\ |||\ x))$$

Our semantics would yield the series of time-extended event structure approximations to $\mathcal{M}[\![\, x \,]\!]$ shown in Figure 11. *(End of example.)*

We will conclude this section by investigating the relation between the denotational semantics for ET-LOTOS as presented in this section and the denotational true concurrency semantics of LOTOS (so, without time) as defined by Langerak [24, 23]. For ET-LOTOS expression P let $\Phi(P)$ denote the corresponding untimed behaviour. This function can easily be defined by induction on the structure of P and is omitted here. In addition, let for time-extended event structure $\Psi = \langle \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ function ϕ select the plain event structure of Ψ, i.e., $\phi(\Psi) = \mathcal{E}$. Let $\mathcal{E} \approx_{tr} \mathcal{E}'$ iff event structures $\mathcal{E}$ and $\mathcal{E}'$ have the same set of event traces, i.e., iff they are equivalent from a partial-order point of view. Then we obtain:

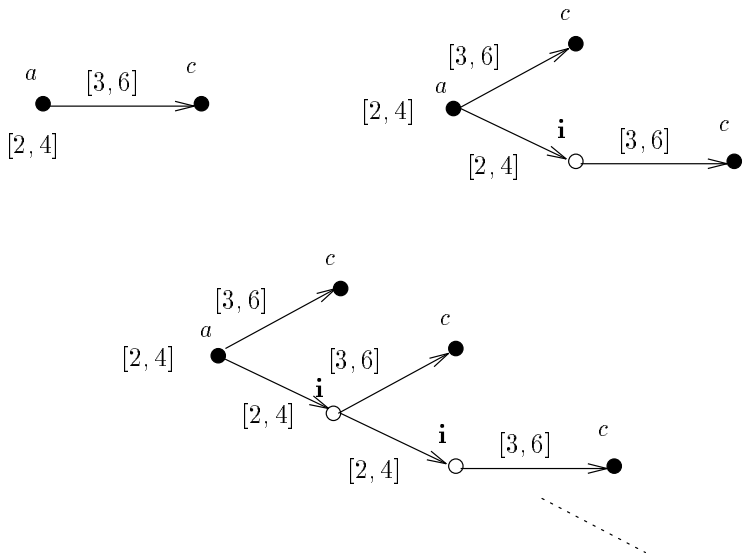


Figure 11: Example fixed point approximations

Theorem 4.27 (Compatibility theorem)
For ET-LOTOS expression P: $\phi(\mathcal{M}[\![P]\!]) \approx_{tr} \mathcal{E}[\![\Phi(P)]\!]$.

Proof:

(sketch.) If we consider the definition of $\mathcal{M}[\![\ ]\!]$ and concentrate on the plain event structure component of $\mathcal{M}[\![\ ]\!]$, i.e., $\phi(\mathcal{M}[\![\ ]\!])$, there are only two syntactical constructs that differ from Langerak's semantics, viz. action-prefix and enabling. In particular, in addition to the untimed case $\mathcal{M}[\![a\,\{T\}\ ;\ P]\!]$ introduces new bundles from $\{ e_a \}$ to all events in $\mathcal{M}[\![P]\!]$ that are time-restricted and non-initial. Similarly, $\mathcal{M}[\![P >> Q]\!]$ introduces in addition to the untimed case bundles between $exit(\mathcal{M}[\![P]\!])$ and the time-restricted non-initial events in $\mathcal{M}[\![Q]\!]$. So, the additional bundles are bundles to *non-initial* events, both for action-prefix and enabling. Consider such bundle $\{ e \} \mapsto e'$ for non-initial e'. Since e' is non-initial, there is some bundle $X \mapsto e'$, $X \neq \{ e \}$. Suppose w.l.o.g. that there is an initial event e'' in X. Then according to the definition of $\mathcal{M}[\![\ ]\!]$ there is also a bundle between $\{ e \}$ and e'' (since e'' is initial). It is now easy to check that in the situation $\{ e \} \mapsto e''$, $\{ e \} \mapsto e'$ and $X \mapsto e'$ with $e'' \in X$, the bundle $\{ e \} \mapsto e'$ is superfluous, since e' causally depends on e'' and e'' on e, which by transitivity of causality implies that e' causally depends on e. (For a full proof of a more general theorem we refer to Theorem 2.44 of Katoen [18].) *(End of proof.)*

As a result of this theorem we can safely state that the causal dependencies are unaffected by the timed constructs in ET-LOTOS, that is, time is added to event structures in a completely orthogonal way.

Example 4.28. Our example is the channel from the multimedia stream specification of Section 2:

$$\begin{aligned} Channel := sourceOut\ ;\ ((&\mathbf{i}\,\{80..92\}\ ;\ sinkIn\,(0)\ ;\ \mathbf{stop} \\ &[]\,\mathbf{i}\,\{0..92\}\ ;\ \mathbf{stop}) \\ &|||\, Channel) \end{aligned}$$

Figure 12 presents the true concurrency model resulting from this behaviour. We have not included the interleaved interpretation since the parallel interleaving of time and action transitions makes it too complicated to draw. In fact, even without showing time transitions the labelled transition system is highly complex.

This example illustrates one of the major benefits of the true concurrency approach: avoidance of state space explosion. In fact, this is a very concise depiction of the behaviour of the channel. *(End of example.)*

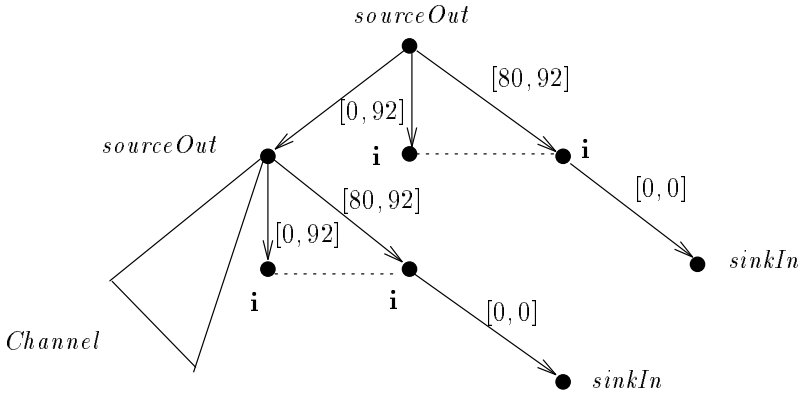


Figure 12: True concurrency model of multimedia stream channel

4.3 Characterising traces of the fixpoint

For plain bundle event structures a very simple relationship exists between the fixed point order and event traces. Specifically, if a bundle event structure $\mathcal{E}_2$ is higher in the fixed point order than $\mathcal{E}_1$ then the event traces of $\mathcal{E}_1$ will be a subset of the event traces of $\mathcal{E}_2$. This is a nice property as it implies that progressing up the fixed point order corresponds to increasing the possible executions of the event structure, where event traces reflect executions.

This property also holds in the simple time-extended event structure model of [18, Chapter 5] and the non-urgent model of [5]. However, once urgency is incorporated the property is lost and this is also the case for the time-extended event structures semantics we have given in this report. As a consequence, the relationship between the fixed point construction and the set of possible event traces is not as straightforward as it is in the untimed and simple timed setting. However, a nice relationship between the event traces of the lub and the traces of approximations to the lub can be given. This result will be similar to Theorem 10.54 of [18], however, we present the main steps in this proof since our definition of timed event sequence is different to that considered in [18]. Before we prove the main result we need a number of lemmas.

Lemma 4.29
Let $\overline{[\sigma]} \subseteq E_1$ and $e \in \mathsf{en}([\sigma])$, then, $\Psi_1 \preceq \Psi_2 \Rightarrow \mathcal{Z}_1(\sigma, e) = \mathcal{Z}_2(\sigma, e)$ [3].

Proof:

Similar to Lemma 10.22 of [18] with single valued timing replaced by interval timing. *(End of proof.)*

[3] Notice that we write $\mathcal{Z}_r$ to denote applying $\mathcal{Z}$ according to time-extended event structure Ψ_r, when not subscripted we are applying according to Ψ.

The next lemma states that any timed event trace of the lub will be a trace of any approximation to the lub if all the events of the trace are already in the approximation. This property ensures that although $\preceq$ is not well behaved as we move up the ordering, in the sense that timed event traces of smaller TESs might not be timed event traces of larger TESs, if we look from the lub down, the ordering is well behaved.

Lemma 4.30
$\sigma \in T(\bigsqcup_i \Psi_i) \Rightarrow (\forall\, k : \overline{[\sigma]} \subseteq E_k \Rightarrow \sigma \in T(\Psi_k))$.

Proof:

Take $\sigma \in T(\bigsqcup_i \Psi_i)$ and let $\bigsqcup_i \Psi_i = \Psi = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ and $\Psi_k = \langle E_k, \rightsquigarrow_k, \mapsto_k, l_k, \mathcal{A}_k, \mathcal{R}_k, \mathcal{I}_k \rangle$ such that $\overline{[\sigma]} \subseteq E_k$. We need to prove that $\sigma \in T(\Psi_k)$. So, let $\sigma = (e_1, t_1)...(e_n, t_n)$ and verify each of the conditions for $\sigma \in T(\Psi_k)$.

1. The event trace and least upper bound conditions for our time-extended event structures correspond to those for bundle event structures. Thus, we can reuse the proof in Theorem 10.24 of [18].

2. This property is not specific to the time-extended event structures being considered. So, it will automatically hold.

3. Using Lemma 4.29 we get $t_i \in \mathcal{Z}_k(\sigma_i, e_i)$ since $\Psi_k \preceq \bigsqcup_i \Psi_i$ and $t_i \in \mathcal{Z}(\sigma_i, e_i)$.

4. $$\begin{aligned}
&\forall\, e \in \mathsf{en}([\sigma_i]) : l(e) = \mathbf{i} \Rightarrow t_i \leqslant f(\mathcal{Z}(\sigma_i, e)) \\
&\qquad \text{where } f = \ \textit{if } e \in \mathcal{I} \textit{ then Min else Max} \\
\Leftrightarrow \quad &\{\ \Psi_1 \preceq \Psi_2 \Rightarrow \mathsf{en}_1([\sigma]) = \mathsf{en}_2([\sigma]) \cap E_1\ ;\ E_1 \subseteq E_2\ ;\ l_1 = l_2 \upharpoonright E_1\ \} \\
&\forall\, e \in \mathsf{en}_k([\sigma_i]) : l_k(e) = \mathbf{i} \Rightarrow t_i \leqslant f(\mathcal{Z}(\sigma_i, e)) \\
&\qquad \text{where } f = \ \textit{if } e \in \mathcal{I} \textit{ then Min else Max} \\
\Leftrightarrow \quad &\{\ \text{Lemma 4.29}; \mathcal{I}_k = \mathcal{I} \cap E_k\ \} \\
&\forall\, e \in \mathsf{en}_k([\sigma_i]) : l_k(e) = \mathbf{i} \Rightarrow t_i \leqslant f(\mathcal{Z}_k(\sigma_i, e)) \\
&\qquad \text{where } f = \ \textit{if } e \in \mathcal{I}_k \textit{ then Min else Max}
\end{aligned}$$

This completes the verification of each of the conditions for $\sigma \in T(\Psi_k)$, as required. *(End of proof.)*

In addition, we introduce the following concept, which is also from [18]. It characterises timed event traces that are present from the nth approximation onwards.

Definition 4.31 (n**-persistent**)
σ is n-persistent if and only if $\exists\, n : (\forall\, j \geqslant n : \sigma \in T(\Psi_j))$. □

The following lemma identifies under what conditions a trace may be a timed event trace of a smaller TES but not of a larger TES.

Lemma 4.32
For $\Psi_1 \preceq \Psi_2$, $\sigma \in T(\Psi_1) \setminus T(\Psi_2) \Rightarrow \exists\, e \in E_2,\ e_i \in \overline{[\sigma]}\ :\ e \in \mathsf{en}_2([\sigma_i]) \wedge l_2(e) = \mathbf{i} \wedge t_i > f(\mathcal{Z}_2(\sigma_i, e))$ where $f = \textit{if } e \in \mathcal{I}_2 \textit{ then Min else Max}$.

Proof:

Assume $\Psi_1 \preceq \Psi_2$, $\sigma \in T(\Psi_1) \setminus T(\Psi_2)$. Our strategy is to consider according to which condition of the definition of T could $\sigma \notin T(\Psi_2)$ hold. It turns out that none of the first three conditions of Definition 3.8 can cause this circumstance to arise.

1. $[\sigma] \in T(\mathcal{E}_1)$ implies $\sigma \in T(\mathcal{E}_2)$, since $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ implies $T(\mathcal{E}_1) \subseteq T(\mathcal{E}_2)$ where $\trianglelefteq$ is the plain bundle event structure fixed point ordering devised by Langerak, see [23].

2. If $\exists\, j$ such that $i < j$ and $t_i > t_j$ then $\sigma \notin T(\Psi_1)$, contradiction.

3. Assume $t_i \in \mathcal{Z}_1(\sigma_i, e_i)$, also we know that $e_i \in \mathsf{en}([\sigma_i])$ and $\overline{[\sigma_i]} \subseteq E_1$. So, we can apply Lemma 4.29 to get that $\mathcal{Z}_1(\sigma_i, e_i) = \mathcal{Z}_2(\sigma_i, e_i) = t_i$.

This verifies that $\sigma \in T(\Psi_1) \setminus T(\Psi_2)$ can only arise because the fifth condition of the definition of T fails, which is negated in the consequence of this lemma. *(End of proof.)*

The following is the result we have been working up to. It states that the timed event traces of the lub are precisely the n-persistent traces of the approximations. This shows that for all timed event traces of $\bigsqcup_i \Psi_i$ there exists a point in the chain from which the trace will appear in all larger approximations.

Theorem 4.33 (Event traces of the fixed point)
$T(\bigsqcup_i \Psi_i) = \bigcup_i \bigcap_{j \geqslant i} T(\Psi_j)$.

Proof:

We consider the two cases: $T(\bigsqcup_i \Psi_i) \subseteq \bigcup_i \bigcap_{j \geqslant i} T(\Psi_j)$ and $T(\bigsqcup_i \Psi_i) \supseteq \bigcup_i \bigcap_{j \geqslant i} T(\Psi_j)$ separately.

$T(\bigsqcup_i \Psi_i) \subseteq \bigcup_i \bigcap_{j \geqslant i} T(\Psi_j)$. This follows directly from Lemma 4.30.

$T(\bigsqcup_i \Psi_i) \supseteq \bigcup_i \bigcap_{j \geqslant i} T(\Psi_j)$. Take $\sigma \in \bigcup_i \bigcap_{j \geqslant i} T(\Psi_j)$, i.e. for some n, $\sigma \in \bigcap_{j \geqslant n} T(\Psi_j)$ and σ is n-persistent. Let $\bigsqcup_i \Psi_i = \Psi = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle = \langle \mathcal{E}, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ and for $r \in \mathbb{N}$, $\Psi_r = \langle E_r, \rightsquigarrow_r, \mapsto_r, l_r, \mathcal{A}_r, \mathcal{R}_r, \mathcal{I}_r \rangle = \langle \mathcal{E}_r, \mathcal{A}_r, \mathcal{R}_r, \mathcal{I}_r \rangle$. We prove the result by contradiction, so we assume that $\sigma \notin T(\bigsqcup_i \Psi_i)$. But, using Lemma 4.32 it follows that there must exist $e \in E$ and $e_i \in \overline{[\sigma]}$ such that

$$e \in \mathsf{en}([\sigma_i]) \wedge l(e) = \mathbf{i} \wedge t_i > f(\mathcal{Z}(\sigma_i, e)) \text{ where } f = \mathit{if}\ \mathcal{I}(e)\ \mathit{then\ Min\ else\ Max}$$

But, since $E = \bigcup_i E_i$, $l = \bigcup_i l_i$ and $\mathcal{I} = \bigcup_i \mathcal{I}_i$ there must exist an m such that $e \in E_m$, $\overline{[\sigma_i]} \subseteq E_m$, $l_m(e) = \mathbf{i}$, $\mathcal{I}_m = \mathcal{I}$ and $\mathcal{Z}_m(\sigma_i, e') = \mathcal{Z}(\sigma_i, e')$ for all $e' \in E_m$. However, this implies that $\sigma \notin T(\Psi_k)$ for all $k \geqslant m$. Hence σ is not n-persistent, i.e. there does not exist an n such that $e \notin \bigcap_{j \geqslant n} T(\Psi_j)$. This gives the required contradiction. *(End of proof.)*

The characterisation of the timed event traces of a least upper bound (under $\preceq$) as provided by Theorem 4.33 is needed when we want to establish consistency (for recursive processes) between the event structure semantics as given in this section and the operational semantics of Section 5. This consistency will be established in Section 6.

5 (Event-Based) Operational Semantics for ET-LOTOS

The operational semantics of ET-LOTOS is based on the interleaving of causally independent events. In Section 6 we investigate to what extent the non-interleaving semantics as presented in the previous section is *consistent* with the interleaving semantics of ET-LOTOS. In order to facilitate this we present in this section an *event-based* operational semantics for ET-LOTOS. We follow the recipe as originally proposed by Boudol & Castellani [4], which was later refined by Langerak [23] in order to show the consistency between his true concurrency semantics of LOTOS and the standard semantics of LOTOS. The idea is to define a transition system in which we keep track of the (times of) occurrences of actions rather than the actions themselves as is usual in structured operational semantics. This results in a *timed-event transition system*. An identical approach has been taken in [19, 21]

In order to deduce events from expressions we subscript each occurrence of an action-prefix and **exit** with an arbitrary but (globally) unique event identifier. We use Greek letters for this purpose. For instance, $a\,\{\,T\,\}\,;\,b\,\{\,T'\,\}\,[]\,b\,\{\,T''\,\}$ becomes $a_\xi\,\{\,T\,\}\,;\,b_\psi\,\{\,T'\,\}\,[]\,b_\chi\,\{\,T''\,\}$. These identifiers play just the rôle of event names. For $P\,|[\Gamma]|\,Q$ new event names are created.

If $e_P \in P$ and $e_Q \in Q$, then possible new names for events in $P \,|[\Gamma]|\, Q$ are $(e_P, *)$ and $(*, e_Q)$ for unsynchronised events and (e_P, e_Q) for synchronised events.

The operational semantics defines two sets of transition relations $\longrightarrow$ and $\rightsquigarrow$. $P \xrightarrow{(e,a)} P'$ denotes that P can perform event e, labelled with action $a \in \mathbf{Act}^{\mathbf{i},\delta}$, and subsequently evolves into P'. $P \overset{d}{\rightsquigarrow} P'$ denotes that P evolves into P' by passing time by d time units ($d \in D_{0\infty}$). During the passage of time no action is performed. The distinction between time and action transitions was introduced by MOLLER & TOFTS [31] for a timed variant of CCS and has also been advocated by NICOLLIN & SIFAKIS [33].

As a subsidiary notion, let $\mathsf{al}(a, P)$ be the set of time instants at which P is allowed to initially perform a. The interpretation of $\mathsf{al}(a, P) = \varnothing$ is that P cannot perform a initially.

Definition 5.1 (Set of initial times to perform an action)
For ET-LOTOS expression P and $a \in \mathbf{Act}^{\mathbf{i},\delta}$, function $\mathsf{al}(a, P)$ is defined as the smallest set satisfying

$$
\begin{array}{rcl}
\mathsf{al}(a, \mathbf{stop}) & \triangleq & \varnothing \\
\mathsf{al}(a, \mathbf{exit}_\xi\,\{\,T\,\}) & \triangleq & \begin{cases} \varnothing & \text{if } a \neq \delta \\ \{\,T\,\} & \text{if } a = \delta \end{cases} \\
\mathsf{al}(a, b_\xi\,\{\,T\,\}\,;\,P) & \triangleq & \begin{cases} \varnothing & \text{if } a \neq b \\ \{\,T\,\} & \text{if } a = b \end{cases} \\
\mathsf{al}(a, P\,[]\,Q) & \triangleq & \mathsf{al}(a,P) \cup \mathsf{al}(a,Q) \\
\mathsf{al}(a, \mathbf{Wait}(d)\,;\,P) & \triangleq & d + \mathsf{al}(a,P) \\
\mathsf{al}(a, P\,[>\,Q) & \triangleq & \mathsf{al}(a,P) \cup \mathsf{al}(a,Q) \\
\mathsf{al}(a, P >> Q) & \triangleq & \begin{cases} \varnothing & \text{if } a = \delta \\ \mathsf{al}(a,P) & \text{if } a \notin \{\,\mathbf{i}, \delta\,\} \\ \mathsf{al}(a,P) \cup \mathsf{al}(\delta,P) & \text{if } a = \mathbf{i} \end{cases} \\
\mathsf{al}(a, P\,|[\Gamma]|\,Q) & \triangleq & \begin{cases} \mathsf{al}(a,P) \cup \mathsf{al}(a,Q) & \text{if } a \notin \Gamma^\delta \\ \mathsf{al}(a,P) \cap \mathsf{al}(a,Q) & \text{if } a \in \Gamma^\delta \end{cases} \\
\mathsf{al}(a, P[H]) & \triangleq & \bigcup\limits_{a=H(b)} \mathsf{al}(b,P) \\
\mathsf{al}(a, \mathbf{hide}\ \Gamma\ \mathbf{in}\ P) & \triangleq & \begin{cases} \varnothing & \text{if } a \in \Gamma \\ \mathsf{al}(a,P) & \text{if } a \notin \Gamma \ \wedge\ a \neq \mathbf{i} \\ \bigcup_{b \in \Gamma^{\mathbf{i}}} \mathsf{al}(b,P) & \text{if } a = \mathbf{i} \end{cases} \\
\mathsf{al}(a, x) & \triangleq & \mathsf{al}(a,Q) \text{ for } x := Q
\end{array}
$$

□

In the sequel let $\mathsf{ma}(A, P)$ for $A \subseteq \mathbf{Act}^{\mathbf{i},\delta}$ abbreviate $Min\,(\bigcup_{a \in A} \mathsf{al}(a, P))$. We simply write $\mathsf{ma}(a, P)$ for $\mathsf{ma}(\{\,a\,\}, P)$. Recall that $Min\,(\varnothing) = \infty$.

Let $\{\,T\,\} \ominus d \triangleq \{\,t - d \mid t \in \{\,T\,\} \ \wedge\ t \geqslant d\,\}$. Notice that the cardinality of $\{\,T\,\}$ and $\{\,T\,\} \ominus d$ are not necessarily the same. In particular, $\{\,T\,\} \ominus d = \varnothing$ if $d > Max(T)$. The relations $\longrightarrow$ and $\rightsquigarrow$ are the smallest relations closed under the inference rules defined below.

Inaction: This behaviour cannot perform any action, but permits any amount of time to

pass while remaining as **stop**.

$$\text{(S)}\ \frac{}{\mathbf{stop} \overset{d}{\rightsquigarrow} \mathbf{stop}}$$

Successful termination: $\mathbf{exit}\{\,T\,\}$ can perform a δ action if sufficient time has elapsed, i.e., if $0 \in \{\,T\,\}$. When it advances time with d time units, it evolves into $\mathbf{exit}\{\,T\,\} \ominus d$.

$$\text{(Ex1)}\ \frac{}{\mathbf{exit}_\xi\{\,T\,\} \xrightarrow{(\xi,\delta)} \mathbf{stop}}\ 0 \in \{\,T\,\} \quad \text{(Ex2)}\ \frac{}{\mathbf{exit}_\xi\{\,T\,\} \overset{d}{\rightsquigarrow} \mathbf{exit}_\xi\{\,T\,\} \ominus d}$$

Observable action-prefix: Like for successful termination, $a\{\,T\,\}\ ;\ P$ can perform a if $0 \in \{\,T\,\}$. It allows any amount of time to pass, with the possibility that a is no longer offered to the environment (viz. if $0 \notin \{\,T\,\} \ominus d$).

$$\text{(AP1)}\ \frac{}{a_\xi\{\,T\,\}\ ;\ P \xrightarrow{(\xi,a)} P}\ 0 \in \{\,T\,\} \quad \text{(AP2)}\ \frac{}{a_\xi\{\,T\,\}\ ;\ P \overset{d}{\rightsquigarrow} a_\xi\{\,T\,\} \ominus d\ ;\ P}$$

Internal action-prefix: The difference with observable action-prefix is that internal actions are urgent. This is reflected by rule (I2) which allows a maximal delay of $Max(T)$ time units only. The only possibility to evolve after such maximal delay is to perform $\mathbf{i}$ (while for the observable case the possibility of further delay also exists, cf. rule (AP2)).

$$\text{(I1)}\ \frac{}{\mathbf{i}_\xi\{\,T\,\}\ ;\ P \xrightarrow{(\xi,\mathbf{i})} P} \qquad 0 \in \{\,T\,\}$$

$$\text{(I2)}\ \frac{}{\mathbf{i}_\xi\{\,T\,\}\ ;\ P \overset{d}{\rightsquigarrow} \mathbf{i}_\xi\{\,T\,\} \ominus d\ ;\ P}\ d \leqslant Max(T)$$

Delay prefix: Behaviour $\mathbf{Wait}(d)\ ;\ P$ will wait for d time units and then evolve into P. This can either be done by waiting for more than d time units in one step (rule (D3)), or by delaying precisely d time units (thus reaching $\mathbf{Wait}(0)\ ;\ P$ using rule (D2)) and then performing some action (rule (D1)). Delays of at most d time units are taken care of by rule (D2).

$$\text{(D1)}\ \frac{P \xrightarrow{(\xi,a)} P'}{\mathbf{Wait}(0)\ ;\ P \xrightarrow{(\xi,a)} P'} \qquad \text{(D2)}\ \frac{}{\mathbf{Wait}(d'+d)\ ;\ P \overset{d}{\rightsquigarrow} \mathbf{Wait}(d')\ ;\ P}$$

$$\text{(D3)}\ \frac{P \overset{d}{\rightsquigarrow} P'}{\mathbf{Wait}(d')\ ;\ P \overset{d+d'}{\rightsquigarrow} P'}$$

Choice: The action rules for [] are equal to those for the untimed case. If P and Q permit the passage of time with some amount then so does their choice $P \,[]\, Q$. Note that the passage of time does not decide the choice between P and Q. (In the 'standard' jargon of NICOLLIN

& SIFAKIS [33] such choice construct is classified as a strong choice; a weak choice allows the passage of time to decide the choice.)

$$\text{(Ch1)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{\begin{array}{l} P \,[]\, Q \xrightarrow{(\xi,a)} P' \\ Q \,[]\, P \xrightarrow{(\xi,a)} P' \end{array}} \qquad \text{(Ch2)}\quad \frac{P \overset{d}{\rightsquigarrow} P', Q \overset{d}{\rightsquigarrow} Q'}{P \,[]\, Q \overset{d}{\rightsquigarrow} P' \,[]\, Q'}$$

Parallel composition: Like for choice, $P \,|[\Gamma]|\, Q$ allows the passage of time with some amount if both components permit this. Components of a parallel composition may perform actions not in the synchronisation set Γ^δ independent of each other, while if both P and Q can participate in a synchronisation action $a \in \Gamma^\delta$ then so can their parallel composition.

$$\text{(PC1)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{\begin{array}{l} P \,|[\Gamma]|\, Q \xrightarrow{((\xi,*),a)} P' \,|[\Gamma]|\, Q \\ Q \,|[\Gamma]|\, P \xrightarrow{((*,\xi),a)} Q \,|[\Gamma]|\, P' \end{array}} \quad a \notin \Gamma^\delta \qquad \text{(PC3)}\quad \frac{P \overset{d}{\rightsquigarrow} P', Q \overset{d}{\rightsquigarrow} Q'}{P \,|[\Gamma]|\, Q \overset{d}{\rightsquigarrow} P' \,|[\Gamma]|\, Q'}$$

$$\text{(PC2)}\quad \frac{P \xrightarrow{(\xi,a)} P', Q \xrightarrow{(\psi,a)} Q'}{P \,|[\Gamma]|\, Q \xrightarrow{((\xi,\psi),a)} P' \,|[\Gamma]|\, Q'} \quad a \in \Gamma^\delta$$

Relabelling: If P can perform action a and evolve into P', then $P[H]$ can perform $H(a)$ and evolve into $P'[H]$. If P allows the passage of time with a certain amount, then so does $P[H]$.

$$\text{(Re1)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{P[H] \xrightarrow{(\xi,H(a))} P'[H]} \qquad \text{(Re2)}\quad \frac{P \overset{d}{\rightsquigarrow} P'}{P[H] \overset{d}{\rightsquigarrow} P'[H]}$$

Disrupt: The action transitions are identical to the untimed case: Q can disrupt P at any point of its execution, except after P has successfully terminated. The passage of time of $P \,[>\, Q$ is defined analogously to the case for choice.

$$\text{(Di1)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{P \,[>\, Q \xrightarrow{(\xi,a)} P' \,[>\, Q} \quad a \neq \delta$$

$$\text{(Di2)}\quad \frac{Q \xrightarrow{(\xi,a)} Q'}{P \,[>\, Q \xrightarrow{(\xi,a)} Q'} \qquad \text{(Di4)}\quad \frac{P \overset{d}{\rightsquigarrow} P', Q \overset{d}{\rightsquigarrow} Q'}{P \,[>\, Q \overset{d}{\rightsquigarrow} P' \,[>\, Q'}$$

$$\text{(Di3)}\quad \frac{P \xrightarrow{(\xi,\delta)} P'}{P \,[>\, Q \xrightarrow{(\xi,\delta)} P'}$$

Hiding: Any action that P can perform, can also be performed by **hide** Γ **in** P whereby actions in set Γ are turned into internal actions, i.e. **i**. Since hidden actions enforce maximal progress, these actions should be performed as soon as possible. This is reflected by rule (H3)

which allows **hide** Γ **in** P to delay with d time units only if there is no hidden action ($a \in \Gamma$) that must be performed earlier.

$$\text{(H1)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{\textbf{hide } \Gamma \textbf{ in } P \xrightarrow{(\xi,a)} \textbf{hide } \Gamma \textbf{ in } P'} \quad a \notin \Gamma$$

$$\text{(H2)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{\textbf{hide } \Gamma \textbf{ in } P \xrightarrow{(\xi,\mathbf{i})} \textbf{hide } \Gamma \textbf{ in } P'} \quad a \in \Gamma$$

$$\text{(H3)}\quad \frac{P \overset{d}{\rightsquigarrow} P'}{\textbf{hide } \Gamma \textbf{ in } P \overset{d}{\rightsquigarrow} \textbf{hide } \Gamma \textbf{ in } P'} \quad d \leqslant \mathsf{ma}(\Gamma, P)$$

Enabling: The action transitions of $P >> Q$ are identical to the untimed case: if P evolves to P' by performing a ($a \neq \delta$) then $P >> Q$ can do the same while evolving into $P' >> Q$, and if P successfully terminates, control is passed to Q. If P can pass time with d time units, then so can $P >> Q$, unless the internal event resulting from the implicit hiding of the successful termination of P (by passing the control to Q) should occur earlier.

$$\text{(En1)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{P >> Q \xrightarrow{(\xi,a)} P' >> Q} \quad a \neq \delta$$

$$\text{(En2)}\quad \frac{P \xrightarrow{(\xi,\delta)} P'}{P >> Q \xrightarrow{(\xi,\mathbf{i})} Q}$$

$$\text{(En3)}\quad \frac{P \overset{d}{\rightsquigarrow} P'}{P >> Q \overset{d}{\rightsquigarrow} P' >> Q} \quad d \leqslant \mathsf{ma}(\delta, P)$$

Process instantiation: The rules for process instantiation are a bit different from the usual case, since we deal with events rather than with actions. First, it is assumed that each process instantiation of x is uniquely identified. Different occurrences of the same process instantiation should produce different event transitions. In addition, event transitions cannot be repeated. For instance, the first event transition of $x := a_\xi \{ 2..7 \} ; x_\pi$ is labeled with (ξ, a); the next time that action a occurs it should be labelled with a label different from ξ, since events need to be unique. These complications are resolved by using an event renaming operator that prefixes all events in a behaviour with a certain occurrence identifier. $\pi(x)$ denotes behaviour x where all event identifiers in x are prefixed with π.

$$\text{(In1)}\quad \frac{Q \xrightarrow{(\xi,a)} Q'}{x_\pi \xrightarrow{(\pi\xi,a)} \pi(Q')} \quad x := Q \qquad \text{(In2)}\quad \frac{Q \overset{d}{\rightsquigarrow} Q'}{x_\pi \overset{d}{\rightsquigarrow} \pi(Q')} \quad x := Q$$

$$\text{(Rn1)}\quad \frac{P \xrightarrow{(\xi,a)} P'}{\pi(P) \xrightarrow{(\pi\xi,a)} \pi(P')} \qquad \text{(Rn2)}\quad \frac{P \overset{d}{\rightsquigarrow} P'}{\pi(P) \overset{d}{\rightsquigarrow} \pi(P')}$$

This completes the event-based operational semantics for ET-LOTOS. The inference rules for $\longrightarrow$ are identical to those of Langerak's event transition system for untimed LOTOS, if we omit the timing information from the expressions (and side conditions of the inference rules). Apart from the fact that we deal with events rather than actions, the main difference between the inference rules defined above and those of LÉONARD & LEDUC [27] is that we use an auxiliary function $\mathsf{ma}(a)P$, whereas they use negative premises. We will now show that our approach is equivalent. In the sequel we use the following (standard) abbreviations:

$$\begin{aligned} P \overset{d}{\rightsquigarrow} &\quad\Leftrightarrow\quad (\exists\, P' : P \overset{d}{\rightsquigarrow} P') \\ P \xrightarrow{a} &\quad\Leftrightarrow\quad (\exists\, e, P' : P \xrightarrow{(e,a)} P') \\ P \not\xrightarrow{a} &\quad\Leftrightarrow\quad \neg\,(P \xrightarrow{a}) \end{aligned}$$

Lemma 5.2
For $d \in D_{0\infty}$ and behaviour P such that $P \overset{d}{\rightsquigarrow}$:

$$d \leqslant \mathsf{ma}(a,P) \Longleftrightarrow \Big((\forall\, P', d' < d : P \overset{d'}{\rightsquigarrow} P' \;\Rightarrow\; P' \not\xrightarrow{a}) \;\wedge\; P \not\xrightarrow{a} \Big)$$

Proof:

By induction on the structure of P.

Base: Assume $P \overset{d}{\rightsquigarrow}$ for $d \in D_{0\infty}$.

1. For $P = \mathbf{stop}$ we have $\mathsf{ma}(a,P) = \infty$ for arbitrary a and the lemma holds, since **stop** cannot perform an action, nor can it evolve into some behaviour that could.
2. For $P = \mathbf{exit}\,\{\,T\,\}$ we distinguish between $a = \delta$ and $a \neq \delta$. For $a \neq \delta$ the proof is analogous to the case for **stop**. Consider $a = \delta$. Then $\mathsf{ma}(a,P) = \mathit{Min}(T)$. '$\Rightarrow$': since $d > 0$ and $d \leqslant \mathit{Min}(T)$ it follows from rule (Ex1) that $P \not\xrightarrow{\delta}$. From rule (Ex2) we infer that $P \overset{d'}{\rightsquigarrow} P'$ with $P' = \mathbf{exit}\,\{\,T\,\} \ominus d'$. For $d' < d \leqslant \mathit{Min}(T)$ this means that $\mathit{Min}(\{\,T\,\} \ominus d') > 0$, so P' cannot perform δ. '$\Leftarrow$': by contradiction. Suppose $d > \mathit{Min}(T)$. Since $d \neq \infty$ this implies $\mathit{Min}(T) \neq \infty$, i.e., $T \neq \varnothing$. For $d' = \mathit{Min}(T)$ it follows from rules (Ex1) and (Ex2) that $P \overset{d'}{\rightsquigarrow} P'$ with $P' = \mathbf{exit}\,\{\,T\,\} \ominus d'$. But, $0 \in \{\,T\,\} \ominus d'$ since $d' = \mathit{Min}(T)$ and $T \neq \varnothing$. So, $P \overset{d'}{\rightsquigarrow} P'$ for $d' < d$ and $P' \xrightarrow{\delta}$. This contradicts the premise for '$\Leftarrow$'.
3. Let $P = b\,\{\,T\,\}\,;Q$. For $a \neq b$ we have $\mathsf{ma}(a,P) = \infty$ and it is clear from rule (AP1) that P can never perform an a initially, so the lemma trivially holds. Consider $a = b$. Then $\mathsf{ma}(a,P) = \mathit{Min}(T)$. For this case the proof is similar to the proof for $\mathbf{exit}\,\{\,T\,\}$, and is omitted here.

Induction step: Assume the lemma holds for Q and R, and assume $P \overset{d}{\rightsquigarrow}$.

1. For $P = Q \,[]\, R$ we derive:

$$\begin{aligned} & d \leqslant \mathsf{ma}(a, Q \,[]\, R) \\ \Leftrightarrow\quad & \{\text{ Definition 5.1; calculus }\} \\ & d \leqslant \mathsf{ma}(a,Q) \;\wedge\; d \leqslant \mathsf{ma}(a,R) \\ \Leftrightarrow\quad & \{\; Q \,[]\, R \overset{d}{\rightsquigarrow} \;\Rightarrow\; Q \overset{d}{\rightsquigarrow} \wedge\; R \overset{d}{\rightsquigarrow};\text{ induction hypothesis }\} \\ & \quad (\forall\, Q', d' < d : Q \overset{d'}{\rightsquigarrow} Q' \;\Rightarrow\; Q' \not\xrightarrow{a}) \;\wedge\; Q \not\xrightarrow{a} \\ & \wedge\; (\forall\, R', d' < d : R \overset{d'}{\rightsquigarrow} R' \;\Rightarrow\; R' \not\xrightarrow{a}) \;\wedge\; R \not\xrightarrow{a} \\ \Leftrightarrow\quad & \{\text{ inference rules (Ch1) and (Ch2) }\} \\ & (\forall\, Q', R', d' < d : Q \,[]\, R \overset{d'}{\rightsquigarrow} Q' \,[]\, R' \;\Rightarrow\; Q' \,[]\, R' \not\xrightarrow{a}) \;\wedge\; Q \,[]\, R \not\xrightarrow{a} \end{aligned}$$

2. For $P = \mathbf{Wait}(d'')\ ;\ Q$ the lemma holds for $d < d''$, since $\mathsf{ma}(a, P) \geqslant d''$ and P can only perform an initial action after delaying at least d'' time units. Consider $d > d''$. Then:

$$
\begin{array}{ll}
& d \leqslant \mathsf{ma}(a, \mathbf{Wait}(d'')\ ;\ Q) \\
\Leftrightarrow & \{\ \text{Definition 5.1; calculus}\ \} \\
& d - d'' \leqslant \mathsf{ma}(a, Q) \\
\Leftrightarrow & \{\ \mathbf{Wait}(d'')\ ;\ Q \overset{d}{\rightsquigarrow}\ \Rightarrow\ Q \overset{d-d''}{\rightsquigarrow}\ ;\ \text{induction hypothesis using } d > d''\ \} \\
& (\forall\, Q', d' < d - d'' : Q \overset{d'}{\rightsquigarrow} Q'\ \Rightarrow\ Q' \overset{a}{\not\longrightarrow})\ \wedge\ Q \overset{a}{\not\longrightarrow} \\
\Leftrightarrow & \{\ \text{inference rules (D1) and (D3)}\ \} \\
& \quad (\forall\, Q', d' + d'' < d : \mathbf{Wait}(d'')\ ;\ Q \overset{d'+d''}{\rightsquigarrow} Q'\ \Rightarrow\ Q' \overset{a}{\not\longrightarrow}) \\
& \wedge\ \mathbf{Wait}(d'')\ ;\ Q \overset{a}{\not\longrightarrow}
\end{array}
$$

3. The proof for $P\ [>\ Q$ goes similar to the proof for $[]$ and is omitted here.

4. Let $P = Q >> R$. Distinguish between $a = \delta$ and $a \neq \delta$. For $a = \delta$ the proof is simple, since $Q >> R$ cannot perform a δ initially, and $\mathsf{al}(\delta, Q >> R)$ equals $\varnothing$. Consider $a \neq \delta$. Then:

$$
\begin{array}{ll}
& d \leqslant \mathsf{ma}(a, Q >> R) \\
\Leftrightarrow & \{\ \text{Definition 5.1}\ \} \\
& d \leqslant \mathsf{ma}(a, Q) \\
\Leftrightarrow & \{\ Q >> R \overset{d}{\rightsquigarrow}\ \Rightarrow\ Q \overset{d}{\rightsquigarrow};\ \text{induction hypothesis}\ \} \\
& (\forall\, Q', d' < d : Q \overset{d'}{\rightsquigarrow} Q'\ \Rightarrow\ Q' \overset{a}{\not\longrightarrow})\ \wedge\ Q \overset{a}{\not\longrightarrow} \\
\Leftrightarrow & \{\ \text{inference rule (En1) using } a \neq \delta\ \} \\
& (\forall\, Q', d' < d : Q \overset{d'}{\rightsquigarrow} Q'\ \Rightarrow\ Q' \overset{a}{\not\longrightarrow})\ \wedge\ Q >> R \overset{a}{\not\longrightarrow} \\
\Leftrightarrow & \{\ \text{inference rules (En1) and (En3); assumption } Q >> R \overset{d}{\rightsquigarrow}\ \} \\
& (\forall\, Q', d' < d : Q >> R \overset{d'}{\rightsquigarrow} Q' >> R\ \Rightarrow\ Q' >> R \overset{a}{\not\longrightarrow})\ \wedge\ Q >> R \overset{a}{\not\longrightarrow}
\end{array}
$$

5. $P = Q\,|[\Gamma]|\,R$. For $a \notin \Gamma$ the proof performs along similar lines as for $[]$ and is omitted here. For $a \in \Gamma$ the proof is rather straightforward and is left to the interested reader.

6. For $P = Q[H]$ we derive:

$$
\begin{array}{ll}
& d \leqslant \mathsf{ma}(a, Q[H]) \\
\Leftrightarrow & \{\ \text{Definition 5.1}\ \} \\
& d \leqslant \mathit{Min}\,(\bigcup_{a=H(b)} \mathsf{al}(b, Q)) \\
\Leftrightarrow & \{\ \text{suppose } b_1, \ldots, b_n \text{ are mapped by } H \text{ on } a\ \} \\
& d \leqslant \mathsf{ma}(b_1, Q)\ \wedge\ \ldots\ \wedge\ d \leqslant \mathsf{ma}(b_n, Q) \\
\Leftrightarrow & \{\ Q[H] \overset{d}{\rightsquigarrow}\ \Rightarrow\ Q \overset{d}{\rightsquigarrow};\ \text{induction hypothesis}\ \} \\
& \forall\, 0 < i \leqslant n : (\forall\, Q', d' < d : Q \overset{d'}{\rightsquigarrow} Q'\ \Rightarrow\ Q' \overset{b_i}{\not\longrightarrow})\ \wedge\ Q \overset{b_i}{\not\longrightarrow} \\
\Leftrightarrow & \{\ \text{inference rule (Re1) and (Re2)}\ \} \\
& (\forall\, Q', d' < d : Q[H] \overset{d'}{\rightsquigarrow} Q'[H]\ \Rightarrow\ Q'[H] \overset{a}{\not\longrightarrow})\ \wedge\ Q[H] \overset{a}{\not\longrightarrow}
\end{array}
$$

7. $P = \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q$. For $a \in \Gamma$ the proof is trivial, since $\mathsf{al}(a, P)$ equals $\varnothing$ and P cannot perform any action $a \in \Gamma$. For $a = \mathbf{i}$ the proof is analogous to relabelling and is omitted here. Consider $a \notin \Gamma$ and $a \neq \mathbf{i}$. Then:

$$
d \leqslant \mathsf{ma}(a, \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q)
$$

$\Leftrightarrow$ { Definition 5.1; $a \notin \Gamma$ and $a \neq \mathbf{i}$ }

$d \leqslant \mathsf{ma}(a, Q)$

$\Leftrightarrow$ { **hide** Γ **in** $Q \overset{d}{\rightsquigarrow}$ $\Rightarrow$ $Q \overset{d}{\rightsquigarrow}$; induction hypothesis }

$(\forall Q', d' < d : Q \overset{d'}{\rightsquigarrow} Q' \;\Rightarrow\; Q' \overset{a}{\not\longrightarrow}) \;\wedge\; Q \overset{a}{\not\longrightarrow}$

$\Leftrightarrow$ { inference rule (H1) }

$(\forall Q', d' < d : Q \overset{d'}{\rightsquigarrow} Q' \;\Rightarrow\; Q' \overset{a}{\not\longrightarrow}) \;\wedge\; \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q \overset{a}{\not\longrightarrow}$

$\Leftrightarrow$ { inference rules (H1) and (H2); assumption **hide** Γ **in** $Q \overset{d}{\rightsquigarrow}$ }

$(\forall Q', d' < d : \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q \overset{d'}{\rightsquigarrow} \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q' \;\Rightarrow\; \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q' \overset{a}{\not\longrightarrow})$
$\wedge\; \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q \overset{a}{\not\longrightarrow}$

8. For $P := Q$ and $\pi(P)$ the proofs are straightforward and are omitted here.

(End of proof.)

As a consequence of Lemma 5.2 the inference rules (H3) and (En3) may be replaced by (H3') and (En3'), respectively:

$$\text{(H3')} \quad \frac{P \overset{d}{\rightsquigarrow} P', \forall a \in \Gamma : ((\forall P'', d' < d : P \overset{d'}{\rightsquigarrow} P'' \;\Rightarrow\; P'' \overset{a}{\not\longrightarrow}) \;\wedge\; P \overset{a}{\not\longrightarrow})}{\mathbf{hide}\ \Gamma\ \mathbf{in}\ P \overset{d}{\rightsquigarrow} \mathbf{hide}\ \Gamma\ \mathbf{in}\ P'}$$

$$\text{(En3')} \quad \frac{P \overset{d}{\rightsquigarrow} P', (\forall P'', d' < d : P \overset{d'}{\rightsquigarrow} P'' \;\Rightarrow\; P'' \overset{\delta}{\not\longrightarrow}) \;\wedge\; P \overset{\delta}{\not\longrightarrow}}{P >> Q \overset{d}{\rightsquigarrow} P' >> Q}$$

These are the inference rules as proposed by Léonard & Leduc [27] and which are currently the basis for the standardisation proposal. Groote [15] has shown that the use of negative premises may introduce two problems: the inference rules may be inconsistent, and secondly it is not obvious how such rules (even if they are consistent) determine a transition relation. Inconsistency means that one can derive from the inference rules that a behaviour can perform an action if and only if it cannot do so. So, if negative premises are used, one has the proof obligation to show that these problems cannot appear. Groote proposes the use of stratification functions to show consistency of a set of rules and shows that once consistency is achieved a particular transition relation can be found. For ET-LOTOS no stratification has been found to our knowledge and the proof that above mentioned problems are avoided is based on Léonard & Leduc [26]. This proof is quite complex. We, therefore, prefer the use of an auxiliary function in side conditions of inference rules. Since this auxiliary function is defined by induction on the structure of expressions, it is obvious that the inference rules are consistent (and determine a transition relation). The obligation to show the equivalence with a negative premise (see Lemma 5.2) is usually not so severe.

We conclude this section with the following properties

Theorem 5.3 (**Time determinism and time additivity**)
For ET-LOTOS behaviours P, Q and R and $d, d' \in D_\infty$ we have:

1. $(P \overset{d}{\rightsquigarrow} Q \;\wedge\; P \overset{d}{\rightsquigarrow} R) \;\Rightarrow\; Q = R$
2. $P \overset{d+d'}{\rightsquigarrow} R \;\Leftrightarrow\; (\exists Q : P \overset{d}{\rightsquigarrow} Q \;\wedge\; Q \overset{d'}{\rightsquigarrow} R)$

Proof:
Straightforward by induction on the structure of P and omitted here. *(End of proof.)*

6 Consistency of Semantics

The purpose of this section is to prove the consistency between the denotational event structure semantics $\mathcal{M}[\![\ ]\!]$ and the event-based operational semantics from Section 5. The consistency criterion we strive for is timed-event trace equivalence. Remark that this is stronger than the usual trace equivalence which is defined on the basis of actions rather than events. Moreover, since the timed transition system is deterministic (apart from the fact that time may progress in different ways, but does so in a confluent manner) timed-event trace equivalence amounts to (timed-event) strong bisimulation equivalence, due to a well-known result by ENGELFRIET [13].

The consistency proof for behaviour P is carried out in the following way. First, we characterise the set of timed traces of P that are generated by the operational semantics and provide a denotational characterisation, denoted $\mathcal{T}[\![P]\!]$, of this set. Subsequently, we prove that the timed traces of $\mathcal{M}[\![P]\!]$ coincide with the traces as defined by $\mathcal{T}[\![P]\!]$.

6.1 Preliminaries

The following notion is needed to characterise the timed traces for parallel composition.

Definition 6.1 (Merging sets of events)
Let S_1 and S_2 be sets of triples of events, actions and a time, and let Γ be a set of action labels ($\Gamma \subseteq \mathbf{Act}$). The set $S_1 \bowtie_\Gamma S_2$ is defined by

$$\begin{aligned} S_1 \bowtie_\Gamma S_2 \triangleq \{\, ((e,e'),a,t) \mid\ & (a \in \Gamma^\delta \ \wedge\ (e,a,t) \in S_1 \ \wedge\ (e',a,t) \in S_2) \ \vee \\ & (a \notin \Gamma^\delta \ \wedge\ (e,a,t) \in S_1 \ \wedge\ e' = *) \ \vee \\ & (a \notin \Gamma^\delta \ \wedge\ e = * \ \wedge\ (e',a,t) \in S_2)\,\} \end{aligned}$$

So, $((e,e'),a,t)$ is a member of $S_1 \bowtie_\Gamma S_2$ if (i) a is a synchronisation event (i.e., $a \in \Gamma^\delta$), $(e,a,t) \in S_1$ and $(e',a,t) \in S_2$ or (ii) a is a non-synchronisation event (i.e., $a \notin \Gamma^\delta$), $(e,a,t) \in S_1$ and $e' = *$ (or similar for $(e',a,t) \in S_2$ and $e = *$). Notice that for case (i) triples of S_1 and S_2 are required to have identical timings.

$(S_1 \bowtie_\Gamma S_2)^*$ consists of all finite time-consistent sequences constructed from elements of the set $S_1 \bowtie_\Gamma S_2$.

Definition 6.2 (Projection)
For $\sigma \in (S_1 \bowtie_\Gamma S_2)^*$ projections $\pi_1(\sigma)$ and $\pi_2(\sigma)$ are defined by:

- $\pi_i(\varepsilon) \triangleq \varepsilon$, for $i = 1, 2$
- $\pi_1(((e,e'),a,t)\ \sigma') \triangleq \begin{cases} \pi_1(\sigma') & \text{if } e = * \\ (e,a,t)\ \pi_1(\sigma') & \text{otherwise} \end{cases}$
- $\pi_2(((e,e'),a,t)\ \sigma') \triangleq \begin{cases} \pi_2(\sigma') & \text{if } e' = * \\ (e',a,t)\ \pi_2(\sigma') & \text{otherwise .} \end{cases}$

□

It is not difficult to check that for $\sigma \in (S_1 \bowtie_\Gamma S_2)^*$ we have that projections $\pi_i(\sigma) \in S_i^*$, for $i = 1, 2$.

Definition 6.3 (**Auxiliary operations on timed traces**)
The following operations on timed event trace σ are defined:

1. For set of actions Γ, $\sigma \setminus \Gamma$ ('σ with Γ hidden') is defined by

   (a) $\varepsilon \setminus \Gamma \triangleq \varepsilon$

   (b) $((e, a, t)\, \sigma') \setminus \Gamma \triangleq \begin{cases} (e, \mathbf{i}, t)\, (\sigma' \setminus \Gamma) & \text{if } a \in \Gamma \\ (e, a, t)\, (\sigma' \setminus \Gamma) & \text{if } a \notin \Gamma \end{cases}$

2. For relabelling function H, $\sigma[H]$ ('σ relabelled with H') is defined by

   (a) $\varepsilon[H] \triangleq \varepsilon$

   (b) $((e, a, t)\, \sigma')[H] \triangleq (e, H(a), t)\, (\sigma'[H])$

3. For $t \in D$, ${}^t[\,\sigma\,]$ ('σ delayed by t') is defined by

   (a) ${}^t[\,\varepsilon\,] \triangleq \varepsilon$

   (b) ${}^t[\,(e, a, t')\, \sigma'\,] \triangleq (e, a, t'+t)\, {}^t[\,\sigma'\,]$

□

Let $\overline{V}$ for V a set of timed event traces denote the set of timed labelled events occurring in a timed trace in V.

Definition 6.4 (**Set of events in a trace**)
For V a set of timed event traces let $\overline{V} \triangleq \{\, s \mid \exists\, \sigma \in V : s \in \overline{\sigma}\,\}$. □

6.2 Operational characterisation of timed traces

For convenience labels of events are omitted in transitions

$$
\begin{array}{rcl}
P \xrightarrow{(e,t)}\!\!\!\!\rightarrow Q & \Leftrightarrow & \exists\, P' : P \overset{t}{\rightsquigarrow} P' \ \wedge\ P' \xrightarrow{e} Q \\
P \xrightarrow{\sigma}\!\!\!\!\rightarrow Q & \Leftrightarrow & \exists\, P_1, \ldots, P_n : \\
 & & P = P_1 \xrightarrow{(e_1,t_1)}\!\!\!\!\rightarrow P_2 \longrightarrow\!\!\!\!\rightarrow \ldots \longrightarrow\!\!\!\!\rightarrow P_{n-1} \xrightarrow{(e_n,t_n)}\!\!\!\!\rightarrow P_n = Q \\
 & & \text{with } \sigma = (e_1, t_1)(e_2, t_1{+}t_2) \ldots (e_n, t_1{+}\ldots{+}t_n) \\
P \xrightarrow{\varepsilon_d}\!\!\!\!\rightarrow Q & \Leftrightarrow & P \overset{d}{\rightsquigarrow} Q, \text{ for some } d \in D_\infty
\end{array}
$$

In the sequel we allow to write $P \overset{0}{\rightsquigarrow}$ in order to avoid case analysis between delays of 0 and more.

Lemma 6.5
$\exists\, R : \mathbf{stop} \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if $\sigma = \varepsilon_d$ and $R = \mathbf{stop}$.

Proof:

Trivial from inference rule (S). *(End of proof.)*

Lemma 6.6
$\exists\, R : \mathbf{exit}_\xi\, \{\, T\,\} \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if

1. $\sigma = \varepsilon_t$ and $R = \mathbf{exit}_\xi \{ T \} \ominus t$, or

2. $\sigma = (\xi, t)$ with $t \in \{ T \}$ and $R = \mathbf{stop}$.

Proof:
Straightforward from inference rules (Ex1) and (Ex2). *(End of proof.)*

Lemma 6.7
For $a \neq \mathbf{i}$ we have $\exists R : a_\xi \{ T \} ; P \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if

1. $\sigma = \varepsilon_t$ and $R = a_\xi \{ T \} \ominus t ; P$, or

2. $\sigma = (\xi, t)^t[\sigma']$ with $t \in \{ T \}$ and $P \xrightarrow{\sigma'}\!\!\!\!\rightarrow R$.

Proof:
The case for $\sigma = \varepsilon$ follows directly from inference rule (AP2). The other case is proven by induction on the length of σ.
Base: for $\sigma = (e, t)$ with $t \in D_\infty$ we derive:

$$
\begin{array}{ll}
 & \exists R : a_\xi \{ T \} ; P \xrightarrow{(e,t)}\!\!\!\!\rightarrow R \\
\Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \} \\
 & \exists R, R' : a_\xi \{ T \} ; P \overset{t}{\rightsquigarrow} R' \ \wedge \ R' \xrightarrow{e} R \\
\Leftrightarrow & \{ \text{ inference rule (AP2) } \} \\
 & \exists R : a_\xi \{ T \} \ominus t ; P \xrightarrow{e} R \\
\Leftrightarrow & \{ \text{ inference rule (AP1) } \} \\
 & 0 \in \{ T \} \ominus t \ \wedge \ e = \xi
\end{array}
$$

Since $P \xrightarrow{\varepsilon}\!\!\!\!\rightarrow P$ and ${}^t[\varepsilon] = \varepsilon$ the lemma follows for this case.
Induction step: assume the lemma holds for $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ and let $\sigma' = \sigma(e, t)$. For convenience, let $\sigma'' = (e_2, t_2) \ldots (e_n, t_n)$. Then we derive:

$$
\begin{array}{ll}
 & \exists R : a_\xi \{ T \} ; P \xrightarrow{\sigma'}\!\!\!\!\rightarrow R \\
\Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \} \\
 & \exists R, R' : a_\xi \{ T \} ; P \xrightarrow{\sigma}\!\!\!\!\rightarrow R' \ \wedge \ R' \xrightarrow{(e, t - \sum t_i)}\!\!\!\!\rightarrow R \\
\Leftrightarrow & \{ \text{ induction hypothesis } \} \\
 & t_1 \in \{ T \} \ \wedge \ e_1 = \xi \ \wedge \ (\exists R, R' : P \xrightarrow{{}^{t_1}[\sigma'']}\!\!\!\!\rightarrow R' \ \wedge \ R' \xrightarrow{(e, t - \sum t_i)}\!\!\!\!\rightarrow R) \\
\Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \} \\
 & t_1 \in \{ T \} \ \wedge \ e_1 = \xi \ \wedge \ (\exists R : P \xrightarrow{{}^{t_1}[\sigma''(e,t)]}\!\!\!\!\rightarrow R)
\end{array}
$$

It now follows that $\sigma = (\xi, t_1)\sigma''(e, t)$ which proves the case. *(End of proof.)*

Lemma 6.8
$\exists R : \mathbf{i}_\xi \{ T \} ; P \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if

1. $\sigma = \varepsilon_t$ and $R = \mathbf{i}_\xi \{ T \} \ominus t ; P$ with $t \leqslant Max(T)$, or

2. $\sigma = (\xi, t)^t[\sigma']$ with $t \in \{ T \}$ and $P \xrightarrow{\sigma'}\!\!\!\!\rightarrow R$.

Proof:
Similar as for the case for observational action-prefix and omitted. *(End of proof.)*

Lemma 6.9
$\exists R : \mathbf{Wait}(d) ; P \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if

1. $\sigma = \varepsilon_t$ and $R = \mathbf{Wait}(d-t)\ ;\ P$ if $t \leqslant d$, and $P \xrightarrow{\varepsilon_{t-d}}\!\!\!\!\rightarrow R$ if $t > d$, or

2. $\sigma \neq \varepsilon$ and $\sigma = {}^{d}[\,\sigma'\,]$ such that $P \xrightarrow{\sigma'}\!\!\!\!\rightarrow R$.

Proof:

The first case follows directly from inference rules (D2) and (D3). The last case is proven by induction on the length of σ, $\sigma \neq \varepsilon$.

Base: for $\sigma = (e,t)$ it is easy to check from the rules (D1) through (D3) that $t \geqslant d$, since $\mathbf{Wait}(d)\ ;\ P$ has to delay for at least d time units before it is able to perform some action. Let $t = d+d'$ for $d' \in D_\infty$.

$$
\begin{array}{ll}
 & \exists\, R : \mathbf{Wait}(d)\ ;\ P \xrightarrow{(e,d+d')}\!\!\!\!\rightarrow R \\
\Leftrightarrow & \{\ \text{definition of } \longrightarrow\!\!\!\!\rightarrow\ \} \\
 & \exists\, R, R' : \mathbf{Wait}(d)\ ;\ P \overset{d+d'}{\rightsquigarrow} R' \ \wedge\ R' \xrightarrow{e} R \\
\Leftrightarrow & \{\ \text{case analysis: } d' = 0 \text{ and } d' > 0\text{; inference rules (D2) and (D3)}\ \} \\
 & (\exists\, R : \mathbf{Wait}(0)\ ;\ P \xrightarrow{e} R) \ \vee\ (\exists\, R, R' : \mathbf{Wait}(d)\ ;\ P \overset{d+d'}{\rightsquigarrow} R' \ \wedge\ R' \xrightarrow{e} R) \\
\Leftrightarrow & \{\ \text{inference rules (D1) and (D3)}\ \} \\
 & (\exists\, R : P \xrightarrow{e} R) \ \vee\ (\exists\, R, R' : P \overset{d'}{\rightsquigarrow} R' \ \wedge\ R' \xrightarrow{e} R) \\
\Leftrightarrow & \{\ \text{definition of } \longrightarrow\!\!\!\!\rightarrow\ \} \\
 & \exists\, R : P \xrightarrow{(e,d')}\!\!\!\!\rightarrow R
\end{array}
$$

Induction step: similar to the case for observational action-prefix and omitted here. *(End of proof.)*

Lemma 6.10

$\exists\, R : P[H] \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if $\sigma = \sigma'[H]$ and $P \xrightarrow{\sigma'}\!\!\!\!\rightarrow R'$ with $R = R'[H]$.

Proof:

By induction on the length of σ. This proof is straightforward from inference rules (Re1) and (Re2) and is omitted. *(End of proof.)*

As a subsidiary notion let $\mathsf{mi}(P)$ denote the minimal time instant at which P is forced to perform an initial internal event. Notice that this is not equal to $\mathsf{ma}(\mathbf{i}, P)$, since this denotes the minimal time at which P is allowed to perform $\mathbf{i}$. For instance, for $P :=$ $\mathbf{i}\,\{\,3..5\,\}\ ;\ \mathbf{stop}\ []\ \mathbf{i}\,\{\,4..8\,\}\ ;\ \mathbf{stop}$, we have $\mathsf{ma}(\mathbf{i}, P) = 3$ whereas we expect $\mathsf{mi}(P)$ to be equal to 5.

Definition 6.11 (Minimal time at which internal action is forced)

Function $\mathsf{mi}(P)$ is defined as

$$
\begin{array}{rcl}
\mathsf{mi}(\mathbf{stop}) & \triangleq & \infty \\
\mathsf{mi}(\mathbf{exit}\,\{\,T\,\}) & \triangleq & \infty \\
\mathsf{mi}(a\,\{\,T\,\}\ ;\ P) & \triangleq & \begin{cases} \infty & \text{if } a \neq \mathbf{i} \\ Max(T) & \text{if } a = \mathbf{i} \end{cases} \\
\mathsf{mi}(P\ \mathsf{op}\ Q) & \triangleq & Min\,(\{\,\mathsf{mi}(P), \mathsf{mi}(Q)\,\}) \text{ for } \mathsf{op} \in \{\ []\ ,\ [>\ ,\ |[]|\ \} \\
\mathsf{mi}(P[H]) & \triangleq & \mathsf{mi}(P) \\
\mathsf{mi}(P >> Q) & \triangleq & Min\,(\{\,\mathsf{mi}(P), \mathsf{ma}(\delta, P)\,\}) \\
\mathsf{mi}(\mathbf{hide}\ \Gamma\ \mathbf{in}\ P) & \triangleq & Min\,(\{\,\mathsf{mi}(P), \mathsf{ma}(\Gamma, P)\,\}) \\
\mathsf{mi}(x) & \triangleq & \mathsf{mi}(Q) \text{ for } x := Q \text{ and } Q \text{ guarded}
\end{array}
$$

□

Lemma 6.12
$\forall\, P, t \in D_\infty : P \overset{t}{\rightsquigarrow} \iff t \leqslant \mathsf{mi}(P)$.

Proof:

By induction on the structure of P. Straightforward and omitted. *(End of proof.)*

Lemma 6.13
$\exists\, R : P \,[]\, Q \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if

1. $\sigma = \varepsilon_t$ and $P \xrightarrow{\sigma}\!\!\!\!\rightarrow P'$ and $Q \xrightarrow{\sigma}\!\!\!\!\rightarrow Q'$ and $R = P' \,[]\, Q'$, or
2. $\sigma = (e,t)\sigma'$ and $P \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ and $t \leqslant \mathsf{mi}(Q)$, or $Q \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ and $t \leqslant \mathsf{mi}(P)$.

Proof:
The first case trivially follows from inference rule (Ch2). The second case is proven by induction on the length of σ.
Base: let $\sigma = (e,t)$. Then we derive:

$$
\begin{array}{ll}
 & \exists\, R : P \,[]\, Q \xrightarrow{(e,t)}\!\!\!\!\rightarrow R \\
\Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \} \\
 & \exists\, R, R' : P \,[]\, Q \overset{t}{\rightsquigarrow} R' \ \wedge\ R' \xrightarrow{e} R \\
\Leftrightarrow & \{ \text{ inference rule (Ch2) } \} \\
 & \exists\, R, P', Q' : P \overset{t}{\rightsquigarrow} P' \ \wedge\ Q \overset{t}{\rightsquigarrow} Q' \ \wedge\ P' \,[]\, Q' \xrightarrow{e} R \\
\Leftrightarrow & \{ \text{ inference rule (Ch1); definition of } \longrightarrow\!\!\!\!\rightarrow \} \\
 & (\exists\, R : P \xrightarrow{(e,t)}\!\!\!\!\rightarrow R \ \wedge\ Q \overset{t}{\rightsquigarrow}) \ \vee\ (\exists\, R : Q \xrightarrow{(e,t)}\!\!\!\!\rightarrow R \ \wedge\ P \overset{t}{\rightsquigarrow}) \\
\Leftrightarrow & \{ \text{ Lemma 6.12 } \} \\
 & (\exists\, R : P \xrightarrow{(e,t)}\!\!\!\!\rightarrow R \ \wedge\ t \leqslant \mathsf{mi}(Q)) \ \vee\ (\exists\, R : Q \xrightarrow{(e,t)}\!\!\!\!\rightarrow R \ \wedge\ t \leqslant \mathsf{mi}(P))
\end{array}
$$

Induction step: straightforward and omitted. *(End of proof.)*

Lemma 6.14
$\exists\, R : P >> Q \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if

1. $\sigma = \varepsilon_t$ and $P \xrightarrow{\varepsilon_t}\!\!\!\!\rightarrow R$ and $t \leqslant \mathsf{ma}(\{\,\delta\,\}, P)$, or
2. $P = P_0 \xrightarrow{(e_1,t_1)}\!\!\!\!\rightarrow \ldots \xrightarrow{(e_n,t_n)}\!\!\!\!\rightarrow P_n$, with $n > 0$ and $\forall\, i : t_i \leqslant \mathsf{ma}(\{\,\delta\,\}, P_i)$ such that either:
   (a) $e_n \neq e_\delta$ and $P >> Q \xrightarrow{(e_1,t_1)}\!\!\!\!\rightarrow \ldots \xrightarrow{(e_n,t_n)}\!\!\!\!\rightarrow R$, or
   (b) $e_n = e_\delta$ and $P >> Q \xrightarrow{(e_1,t_1)}\!\!\!\!\rightarrow \ldots \xrightarrow{(e_\delta,t_n)}\!\!\!\!\rightarrow Q \xrightarrow{\sigma'}\!\!\!\!\rightarrow R$.

Proof:
The first case follows directly from inference rule (En3). The proof of the second case is by induction on the length of σ.
Base: let $\sigma = (e_1,t_1)$. Then we infer:

$$
\begin{array}{ll}
 & \exists\, R : P >> Q \xrightarrow{(e_1,t_1)}\!\!\!\!\rightarrow R \\
\Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \text{; inference rule (En3) } \} \\
 & \exists\, R, R' : P >> Q \overset{t_1}{\rightsquigarrow} R' >> Q \ \wedge\ R' >> Q \xrightarrow{e_1} R \\
\Leftrightarrow & \{ \text{ inference rule (En3) } \}
\end{array}
$$

$$
\begin{array}{ll}
 & \exists\, R, R' : P \stackrel{t_1}{\rightsquigarrow} R' \ \wedge\ t_1 \leqslant \mathsf{ma}(\{\,\delta\,\}, P) \ \wedge\ R' >> Q \xrightarrow{e_1} R \\
\Leftrightarrow & \{ \text{ inference rules (En1) and (En2) } \} \\
 & \quad \exists\, R, R' : P \stackrel{t_1}{\rightsquigarrow} R' \ \wedge\ t_1 \leqslant \mathsf{ma}(\{\,\delta\,\}, P) \ \wedge\ R' >> Q \xrightarrow{e_1} R \\
 & \wedge\ (\exists\, R'' : R' \xrightarrow{e_1} R'' \ \wedge\ (e_1 \neq e_\delta \ \wedge\ R = R'' >> Q) \ \vee\ (e_1 = e_\delta \ \wedge\ R = Q)) \\
\Leftrightarrow & \{ \text{ definition of } \twoheadrightarrow \} \\
 & \quad \exists\, R, R'' : P \xrightarrow{(e_1, t_1)} R'' \ \wedge\ t_1 \leqslant \mathsf{ma}(\{\,\delta\,\}, P) \\
 & \wedge\ ((e_1 \neq e_\delta \ \wedge\ R = R'' >> Q) \ \vee\ (e_1 = e_\delta \ \wedge\ R = Q))
\end{array}
$$

Induction step: straightforward, but quite tedious. This proof is left to the interested reader. *(End of proof.)*

Lemma 6.15
$\exists\, R : \mathbf{hide}\ \Gamma\ \mathbf{in}\ P \xrightarrow{\sigma} R$ if and only if

1. $\sigma = \varepsilon_t$ and $t \leqslant \mathsf{ma}(\Gamma, P)$, or
2. $\sigma = \sigma' \setminus \Gamma$ such that $P \xrightarrow{\sigma'} R'$ and $R = \mathbf{hide}\ \Gamma\ \mathbf{in}\ R'$ where if $P_i \xrightarrow{(e_{i+1}, t_{i+1})} P_{i+1}$ then $t_{i+1} \leqslant \mathsf{ma}(\Gamma, P_i)$ for $0 \leqslant i < n$.

Proof:
The first case follows directly from inference rule (H3). For the other case the proof is sketched below. Let $P = P_0$, $R = \mathbf{hide}\ \Gamma\ \mathbf{in}\ P_n$, and $0 \leqslant i < n$. Then:

$$
\begin{array}{ll}
 & \exists\, P_i : \mathbf{hide}\ \Gamma\ \mathbf{in}\ P_i \xrightarrow{(e_{i+1}, t_{i+1})} \mathbf{hide}\ \Gamma\ \mathbf{in}\ P_{i+1} \\
\Leftrightarrow & \{ \text{ definition of } \twoheadrightarrow \} \\
 & \exists\, P_i, P_i' : \mathbf{hide}\ \Gamma\ \mathbf{in}\ P_i \stackrel{t_{i+1}}{\rightsquigarrow} \mathbf{hide}\ \Gamma\ \mathbf{in}\ P_i' \wedge \mathbf{hide}\ \Gamma\ \mathbf{in}\ P_i' \xrightarrow{e_{i+1}} \mathbf{hide}\ \Gamma\ \mathbf{in}\ P_{i+1} \\
\Leftrightarrow & \{ \text{ inference rules (H1), (H2) and (H3) } \} \\
 & \exists\, P_i, P_i' : P_i \stackrel{t_{i+1}}{\rightsquigarrow} P_i' \ \wedge\ t_{i+1} \leqslant \mathsf{ma}(\Gamma, P_i) \ \wedge\ P_i' \xrightarrow{e_{i+1} \setminus \Gamma} P_{i+1} \\
\Leftrightarrow & \{ \text{ definition of } \twoheadrightarrow \} \\
 & \exists\, P_i : P_i \xrightarrow{(e_{i+1} \setminus \Gamma, t_{i+1})} P_{i+1} \ \wedge\ t_{i+1} \leqslant \mathsf{ma}(\Gamma, P_i)
\end{array}
$$

(End of proof.)

Lemma 6.16
$\exists\, R : P\ [>\ Q \xrightarrow{\sigma} R$ if and only if

1. $\sigma = \varepsilon_t$ and $P \xrightarrow{\sigma} P'$, $Q \xrightarrow{\sigma} Q'$ and $R = P'\ [>\ Q'$, or
2. $P\ [>\ Q = P_0\ [>\ Q_0 \xrightarrow{(e_1, t_1)} \ldots \xrightarrow{(e_{n-1}, t_{n-1})} P_{n-1}\ [>\ Q_{n-1} \xrightarrow{(e_\delta, t_n)} P_n = R$ with $n > 0$ and $t_{i+1} \leqslant \mathsf{mi}(Q_i)$ for $0 \leqslant i < n$[4], or
3. $P\ [>\ Q = P_0\ [>\ Q_0 \xrightarrow{(e_1, t_1)} \ldots \xrightarrow{(e_n, t_n)} P_n\ [>\ Q_n \xrightarrow{(e_1', t_1')} Q_1' \xrightarrow{(e_2', t_2')} \ldots \xrightarrow{(e_m', t_m')} Q_m' = R$ with $m+n > 0$ such that $e_n \neq e_\delta$ with $t_{i+1} \leqslant \mathsf{mi}(Q_i)$ for $0 \leqslant i < n$ and $t_1' \leqslant \mathsf{mi}(P_n)$.

Proof:
The first case follows directly from inference rule (Di4). For the other two cases the proof is sketched below. Let $\sigma = (e_1, t_1) \ldots (e_n, t_n)(e_1', t_1') \ldots (e_m', t_m')$ with $n, m \geqslant 0$ and $n+m > 0$. Distinguish between the following cases: (1) $n = 0$, (2) $m = 0$, and (3) $n, m > 0$. For these cases we have:

[4] It is not difficult to check that this constraint is identical to $\sum t_i \leqslant \mathsf{mi}(Q)$.

1. let $Q_m = R$ and $Q = Q_0$. Then we infer for all $0 \leqslant i < n$:

$$\begin{array}{ll} & \exists\, Q_i : P \,[\!>\, Q_0 \xrightarrow{(e'_1,t'_1)}\!\!\!\!\rightarrow\; Q_1 \xrightarrow{(e'_2,t'_2)}\!\!\!\!\rightarrow \ldots \xrightarrow{(e'_m,t'_m)}\!\!\!\!\rightarrow\; Q_m \\ \Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \} \\ & \exists\, Q_i, P', Q'_0 : P \,[\!>\, Q_0 \overset{t'_1}{\rightsquigarrow} P' \,[\!>\, Q'_0 \xrightarrow{e'_1} Q_1 \wedge Q_1 \xrightarrow{(e'_2,t'_2)}\!\!\!\!\rightarrow \ldots \xrightarrow{(e'_m,t'_m)}\!\!\!\!\rightarrow\; Q_m \\ \Leftrightarrow & \{ \text{ inference rules (Di4) and (Di2); definition of } \longrightarrow\!\!\!\!\rightarrow \} \\ & \exists\, Q_i, P' : P \overset{t'_1}{\rightsquigarrow} P' \;\wedge\; Q_0 \xrightarrow{(e'_1,t'_1)}\!\!\!\!\rightarrow \ldots \xrightarrow{(e'_m,t'_m)}\!\!\!\!\rightarrow\; Q_m \\ \Leftrightarrow & \{ \text{ Lemma 6.12 } \} \\ & \exists\, Q_i : t'_1 \leqslant \mathsf{mi}(P) \;\wedge\; Q_0 \xrightarrow{(e'_1,t'_1)}\!\!\!\!\rightarrow \ldots \xrightarrow{(e'_m,t'_m)}\!\!\!\!\rightarrow\; Q_m \end{array}$$

2. assume for the sake of convenience that $e_n \neq e_\delta$. It is easy to check that this implies that $e_i \neq e_\delta$ for all i. Let $P \,[\!>\, Q = P_0 \,[\!>\, Q_0$ and $R = P_n \,[\!>\, Q_n$. Then we infer for all $0 \leqslant i < n$:

$$\begin{array}{ll} & \exists\, P_i, Q_i : P_i \,[\!>\, Q_i \xrightarrow{(e_{i+1},t_{i+1})}\!\!\!\!\rightarrow\; P_{i+1} \,[\!>\, Q_{i+1} \\ \Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \} \\ & \exists\, P_i, P'_i, Q_i, Q'_i : P_i \,[\!>\, Q_i \overset{t_{i+1}}{\rightsquigarrow} P'_i \,[\!>\, Q'_i \;\wedge\; P'_i \,[\!>\, Q'_i \xrightarrow{e_{i+1}} P_{i+1} \,[\!>\, Q_{i+1} \\ \Leftrightarrow & \{ \text{ inference rule (Di4) } \} \\ & \exists\, P_i, P'_i, Q_i, Q'_i : P_i \overset{t_{i+1}}{\rightsquigarrow} P'_i \wedge Q_i \overset{t_{i+1}}{\rightsquigarrow} Q'_i \wedge P'_i \,[\!>\, Q'_i \xrightarrow{e_{i+1}} P_{i+1} \,[\!>\, Q_{i+1} \\ \Leftrightarrow & \{ \text{ inference rule (Di2) using that } e_i \neq e_\delta \} \\ & \exists\, P_i, P'_i, Q_i, Q'_i : P_i \overset{t_{i+1}}{\rightsquigarrow} P'_i \;\wedge\; Q_i \overset{t_{i+1}}{\rightsquigarrow} Q'_i \;\wedge\; P'_i \xrightarrow{e_{i+1}} P_{i+1} \\ \Leftrightarrow & \{ \text{ definition of } \longrightarrow\!\!\!\!\rightarrow \text{; Lemma 6.12 } \} \\ & \exists\, P_i, Q_i : P_i \xrightarrow{(e_{i+1},t_{i+1})}\!\!\!\!\rightarrow\; P_{i+1} \;\wedge\; t_{i+1} \leqslant \mathsf{mi}(Q_i) \end{array}$$

   For $e_n = e_\delta$ the proof is analogous, except that in the last step P_n is reached rather than $P_n \,[\!>\, Q_n$.

3. The proof for this case is in fact a combination of the derivation given above for cases (1) and (2) and we omit this proof here.

(End of proof.)

Lemma 6.17
$\exists\, R : P \,|[\Gamma]|\, Q \xrightarrow{\sigma}\!\!\!\!\rightarrow R$ if and only if $P \xrightarrow{\pi_1(\sigma)}\!\!\!\!\rightarrow P'$, $Q \xrightarrow{\pi_2(\sigma)}\!\!\!\!\rightarrow Q'$ and $R = P' \,|[\Gamma]|\, Q'$.

Proof:

The proof is straightforward (but tedious) by induction on the length of σ, using the definition of $\bowtie_\Gamma$.

(End of proof.)

6.3 A denotational event-trace semantics

Let $\mathsf{aft}(P, \sigma)$ denote behaviour P' that is obtained from P by performing σ. Remark that $\mathsf{aft}(P, \sigma)$ leads in our case to a unique state since we deal with event traces rather than the more common action traces.

Definition 6.18 (Denotational event trace semantics)
For ET-LOTOS behaviour P the set $\mathcal{T}[\![\, P \,]\!]$ of timed event traces is defined by:

1. $\mathcal{T}[\![\, \mathbf{stop} \,]\!] \triangleq \{\, \varepsilon \,\}$

2. $\mathcal{T}[\![\, \mathbf{exit}_\xi \{ T \} \,]\!] \triangleq \{ \varepsilon \} \cup \{ (\xi, t) \mid t \in \{ T \} \}$

3. $\mathcal{T}[\![\, a_\xi \{ T \} ; P \,]\!] \triangleq \{ \varepsilon \} \cup \{ (\xi, t)^t[\sigma] \mid t \in \{ T \}, \sigma \in \mathcal{T}[\![P]\!] \}$

4. $\mathcal{T}[\![\, \mathbf{Wait}(d) ; P \,]\!] \triangleq \{ {}^d[\sigma] \mid \sigma \in \mathcal{T}[\![P]\!] \}$

5. $\mathcal{T}[\![\, P \,[]\, Q \,]\!] \triangleq \{ (e,t)\sigma \in \mathcal{T}[\![P]\!] \mid t \leqslant \mathsf{mi}(Q) \} \cup \{ (e,t)\sigma \in \mathcal{T}[\![Q]\!] \mid t \leqslant \mathsf{mi}(P) \} \cup \{ \varepsilon \}$

6. $\mathcal{T}[\![\, P[H] \,]\!] \triangleq \{ \sigma[H] \mid \sigma \in \mathcal{T}[\![P]\!] \}$

7. $\mathcal{T}[\![\, \mathbf{hide}\ \Gamma\ \mathbf{in}\ P \,]\!] \triangleq \{ \sigma \setminus \Gamma \mid \sigma \in \mathcal{T}[\![P]\!], \forall\, i : t_i - t_{i-1} \leqslant \mathsf{ma}(\Gamma, \mathsf{aft}(P, \sigma_i)) \}$

8. $\mathcal{T}[\![\, P >> Q \,]\!] \triangleq \{ \sigma \in \mathcal{T}[\![P]\!] \mid \sigma \neq \sigma'(e_\delta, t), \forall\, i : t_i - t_{i-1} \leqslant \mathsf{ma}(\delta, \mathsf{aft}(P, \sigma_i)) \}$
$\quad \cup \{ \sigma(e_{\mathbf{i}}, t)^t[\sigma'] \mid \sigma(e_\delta, t) \in \mathcal{T}[\![P]\!], \sigma' \in \mathcal{T}[\![Q]\!], \forall\, i : t_i - t_{i-1} \leqslant \mathsf{ma}(\delta, \mathsf{aft}(P, \sigma_i)) \}$

9. $\mathcal{T}[\![\, P \,[> Q \,]\!] \triangleq \quad \{ \varepsilon \} \cup \{ \sigma(e_\delta, t) \in \mathcal{T}[\![P]\!] \mid t \leqslant \mathsf{mi}(Q) \}$
$\quad \cup \{ (e,t)\sigma \in \mathcal{T}[\![Q]\!] \mid t \leqslant \mathsf{mi}(P) \}$
$\quad \cup \{ \sigma_1\sigma_2 \mid \sigma_1 = \sigma'(e,t) \in \mathcal{T}[\![P]\!], e \neq e_\delta, t \leqslant \mathsf{mi}(Q),$
$\qquad \sigma_2 = (e', t')\sigma'' \in \mathcal{T}[\![Q]\!], 0 \leqslant t' - t \leqslant \mathsf{mi}(\mathsf{aft}(P, \sigma_1)) \}$

10. $\mathcal{T}[\![\, P \,|[\Gamma]|\, Q \,]\!] \triangleq \{ \sigma \in (\overline{\mathcal{T}[\![P]\!]} \bowtie_\Gamma \overline{\mathcal{T}[\![Q]\!]})^* \mid \pi_1(\sigma) \in \mathcal{T}[\![P]\!], \pi_2(\sigma) \in \mathcal{T}[\![Q]\!] \}$

□

6.4 Consistency between operational and denotational semantics

Lemma 6.19
For all P we have: $\mathcal{T}[\![P]\!] = \{ \sigma \mid \exists\, Q : P \xrightarrow{\sigma\!\!\rightarrow} Q \}$.

Proof:

Directly from Definition 6.18 and Lemmata 6.2 through 6.17. *(End of proof.)*

We will use the following small lemmas in proving the consistency result of this subsection.

Lemma 6.20
Let $\Psi = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ then $\mathsf{en}(\varepsilon) = \mathit{init}(\Psi)$.

Proof:

$\mathsf{en}(\varepsilon) = \mathsf{sat}(\varepsilon) \setminus (\mathsf{cfl}(\varepsilon) \cup \overline{\varepsilon})$

$=$ { definition of sat and cfl }
$\mathsf{sat}(\varepsilon) \setminus \varnothing = \{ e \in E \mid \forall\, X \subseteq E : X \mapsto e \ \Rightarrow\ (X \cap \varnothing) \neq \varnothing \}$

$=$ { calculus }
$\{ e \in E \mid \forall\, X \subseteq E : X \mapsto e \ \Rightarrow\ \text{false} \}$

$=$ { calculus }
$\{ e \in E \mid \neg\, (\exists\, X \subseteq E : X \mapsto e) \}$

$=$ { definition of *init* }
$\mathit{init}(\Psi)$ *(End of proof.)*

Lemma 6.21
Let $\mathcal{M}[\![P]\!] = \Psi = \langle E, \leadsto, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ and $\sigma = (e_1, t_1) \ldots (e_n, t_n) \in T(\Psi)$ then, $\forall\, t \in D$:

$$\neg\, (\exists\, e \in \mathsf{en}([\sigma]) : l(e) = \mathbf{i} \ \wedge\ t > f(\mathcal{Z}(\sigma, e))) \iff \mathsf{aft}(P, \sigma) \stackrel{t-t_n}{\leadsto}$$

where $f =$ *if* $e \in \mathcal{I}$ *then Min else Max*.

Proof:

We sketch the proof of this lemma.

($\Longrightarrow$) Prove by contrapositive. So, assume $\neg(\mathsf{aft}(P, \sigma) \stackrel{t-t_n}{\leadsto})$. Now from an inspection of the operational semantics rules of Section 5 this is only possible if some initial component behaviour of $\mathsf{aft}(P, \sigma)$ is either of the form:

1. $\mathbf{i}_\xi\{T\}$; Q and $t{-}t_n > Max(T)$; or
2. **hide** Γ **in** Q and $t{-}t_n > \mathsf{ma}(\Gamma, Q)$; or
3. $Q >> R$ and $t{-}t_n > \mathsf{ma}(\delta, Q)$;

(in addition, the delay operator, **Wait**(d) ; Q, can affect the time at which a behaviour is prevented from passing time, but we could use an inductive argument on the structure of Q to handle this situation. So, w.l.o.g. we ignore the case here.) We consider each of the above possibilities in turn:

1. This implies that ξ is a suitable event to give us the result we want, since $l(\xi) = \mathbf{i}$, $\xi \in \mathsf{en}([\sigma])$ and $t{-}t_n > Max(T) \Rightarrow t > Max(T) + t_n = f(\mathcal{Z}(\sigma, \xi))$.
2. $t{-}t_n > \mathsf{ma}(\Gamma, Q)$ implies that there exists an initial action of Q that is in Γ which becomes (immediately) urgent after less than $t{-}t_n$ time units. Now this action will induce an event e in E, that will be enabled after σ (since it is initial in Q), such that $l(e) = \mathbf{i}$ (because the action is in Γ) and $t{-}t_n > \mathsf{ma}(\Gamma, Q) \Rightarrow t > \mathsf{ma}(\Gamma, Q) + t_n \geqslant f(\mathcal{Z}(\sigma, e))$.
3. This case is similar to the last.

($\Longleftarrow$) Prove by contrapositive. So assume $\exists\, e \in \mathsf{en}([\sigma]) : l(e) = \mathbf{i} \ \wedge\ t > f(\mathcal{Z}(\sigma, e))$, but this implies that there exists an initial internal action in $\mathsf{aft}(B, \sigma)$ that becomes urgent before t time units have past. By again considering under what circumstances ET-LOTOS specifications can allow time to pass we can verify that $\neg(\mathsf{aft}(P, \sigma) \stackrel{t-t_n}{\leadsto})$, as required. *(End of proof.)*

Lemma 6.22
Let $\mathcal{M}[\![P]\!] = \Psi = \langle E, \leadsto, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle, \sigma = (e_1, t_1) \ldots (e_n, t_n) \in T(\Psi)$, $\Gamma \subseteq \mathbf{Act}$ and actions in Γ are immediate. Then, $\forall\, t \in D$:

$$(\exists\, e \in \mathsf{en}([\sigma]) : l(e) \in \Gamma \ \wedge\ t > \mathit{Min}(\mathcal{Z}(\sigma, e))) \iff t{-}t_n > \mathsf{ma}(\Gamma, \mathsf{aft}(P, \sigma))$$

Proof:

We sketch the proof of this lemma.

($\Longrightarrow$) Assume $(\exists\, e \in \mathsf{en}([\sigma]) : l(e) \in \Gamma \ \wedge\ t > \mathit{Min}(\mathcal{Z}(\sigma, e)))$. Thus, after σ there exists an urgent event that must occur before t time units have past. However, if there exists an event, e, that is enabled after σ that is labelled $a \in \Gamma$ then a must be initially offered by $\mathsf{aft}(P, \sigma)$. Furthermore, since actions in Γ are immediate we can see that $t_n + \mathit{Min}(\mathsf{al}(a, \mathsf{aft}(P, \sigma))) = \mathit{Min}(\mathcal{Z}(\sigma, e))$, which implies that $t > \mathit{Min}(\mathcal{Z}(\sigma, e)) = t_n + \mathsf{ma}(\Gamma, \mathsf{aft}(P, \sigma))$ which implies that $t - t_n > \mathsf{ma}(\Gamma, \mathsf{aft}(P, \sigma))$, as required.

($\Longleftarrow$) Assume $t - t_n > \mathsf{ma}(\Gamma, \mathsf{aft}(P, \sigma))$ which means that $\exists\, a \in \Gamma : t - t_n > \mathit{Min}(\mathsf{al}(a, \mathsf{aft}(P, \sigma)))$. Since a is initially offered in $\mathsf{aft}(P, \sigma)$, we know that there must exist a corresponding event e that is

enabled, i.e. $e \in \mathsf{en}([\sigma])$, such that $l(e) \in \Gamma$ and $t > \mathit{Min}(\mathsf{al}(\mathrm{a}, \mathsf{aft}(P, \sigma))) + t_n = \mathit{Min}(\mathcal{Z}(\sigma, e))$ which implies that $t > \mathit{Min}(\mathcal{Z}(\sigma, e))$, as required. *(End of proof.)*

Now to the main result of the subsection. This lemma verifies that the set of event traces derived by the denotational semantics of Definition 6.18 is the same as the set of event traces that are arrived at via the derivation of timed event traces (Definition 3.8) from the corresponding time-extended event structure.

Lemma 6.23
$\forall P \in \text{ET-LOTOS} : T(\mathcal{M}[\![P]\!]) = \mathcal{T}[\![P]\!]$

Proof:

We prove this lemma by induction on the structure of the behaviour expression P.

Base Case. If $P = \mathbf{stop}$ then $T(\mathcal{M}[\![\mathbf{stop}]\!]) = T(-) = \{ \varepsilon \} = \mathcal{T}[\![\mathbf{stop}]\!]$. Alternatively, if $P = \mathbf{exit}_\xi \{ T \}$ then by considering the rules for generating event traces from time-extended event structures we can see that,

$$T(\mathcal{M}[\![\mathbf{exit}_\xi \{T\}]\!]) = T(\langle \{\xi\}, \varnothing, \varnothing, \{(\xi, \delta)\}, \{(\xi, \{T\})\}, \varnothing, \varnothing \rangle) =$$
$$\{\varepsilon\} \cup \{(\xi, t) \mid t \in \{T\}\} = \mathcal{T}[\![\mathbf{exit}_\xi \{T\}]\!]$$

Inductive Step. Assume the result for Q and R. In addition, let $\Psi = \mathcal{M}[\![P]\!] = \langle E, \rightsquigarrow, \mapsto, l, \mathcal{A}, \mathcal{R}, \mathcal{I} \rangle$ and $\Psi_Q = \mathcal{M}[\![Q]\!] = \langle E_Q, \rightsquigarrow_Q, \mapsto_Q, l_Q, \mathcal{A}_Q, \mathcal{R}_Q, \mathcal{I}_Q \rangle$; define Ψ_R analogously. Consider in turn each of the possible syntactic forms for P.

1. *Action Prefix.* Let $P = a_\xi\{T\} \; ; Q$. The events of Ψ are as follows $E = E_Q \cup \{\xi\}$ and bundles have been added as follows: $\mapsto = \mapsto_Q \cup (\{\{\xi\}\} \times \mathsf{rin}(\Psi_Q))$. Thus, we can see that the non-empty timed event traces of Ψ will be of the form:

   $$(\xi, t)^t[\sigma]$$

   where σ is a timed event trace of Ψ_Q and $t \in \mathcal{A}(\xi) = \{T\}$ (notice that the last condition of Definition 3.8 does not apply since $\mathsf{en}(\varepsilon)$ is the singleton set just containing ξ). Thus, we can derive the following:

   $$\begin{aligned}
   & T(\mathcal{M}[\![a_\xi\{T\} \; ; Q]\!]) \\
   = \quad & \{ \text{ by the argument just made } \} \\
   & \{\varepsilon\} \cup \{(\xi, t)^t[\sigma] \mid t \in \{T\} \ \wedge \ \sigma \in T(\Psi_Q)\} \\
   = \quad & \{ \text{ inductive hypothesis } \} \\
   & \{\varepsilon\} \cup \{(\xi, t)^t[\sigma] \mid t \in \{T\} \ \wedge \ \sigma \in \mathcal{T}[\![Q]\!]\} \\
   = \quad & \{ \text{ Definition 6.18 } \} \\
   & \mathcal{T}[\![a_\xi\{T\} \; ; Q]\!]
   \end{aligned}$$

2. *Delay.* Let $P = \mathbf{Wait}(d) \; ; Q$. The only component of Ψ_Q that changes in $\mathcal{M}[\![\mathbf{Wait}(d) \; ; Q]\!]$ is the delay constraint. Specifically, $\mathcal{A} = (+d) \circ \mathcal{A}_Q$. Consequently it follows that the traces of Ψ have the form ${}^d[\sigma]$ where $\sigma \in T(\Psi_Q)$. Thus, we can derive the following:

   $$\begin{aligned}
   & T(\mathcal{M}[\![\mathbf{Wait}(d) \; ; Q]\!]) \\
   = \quad & \{ \text{ by the argument just made } \} \\
   & \{ {}^d[\sigma] \mid \sigma \in T(\Psi_Q) \} \\
   = \quad & \{ \text{ inductive hypothesis } \} \\
   & \{ {}^d[\sigma] \mid \sigma \in \mathcal{T}[\![Q]\!] \} \\
   = \quad & \{ \text{ Definition 6.18 } \} \\
   & \mathcal{T}[\![\mathbf{Wait}(d) \; ; Q]\!]
   \end{aligned}$$

3. *Choice.* Let $P = Q \,[]\, R$. The effect of $\mathcal{M}[\![Q \,[]\, R]\!]$ is to place initial events of Ψ_Q and Ψ_R mutually in conflict. Consequently, a timed event trace is in $\mathcal{T}[\![P]\!]$ if it is in either $\mathcal{T}[\![Q]\!]$ or $\mathcal{T}[\![R]\!]$ unless an initial event is prevented from happening by being placed in conflict with an urgent event. For example, if e_Q and e_R are initial events of Ψ_Q respectively Ψ_R, such that $l_Q(e_Q) = \mathbf{i}$, e_R is excluded from happening at time t if $t > f(\mathcal{A}_Q(e_Q))$ where $f = \mathit{if}\ e_Q \in \mathcal{I}_Q\ \mathit{then}\ \mathit{Min}\ \mathit{else}\ \mathit{Max}$. Thus, we can derive the following (where $f_Q = \mathit{if}\ e' \in \mathcal{I}_Q\ \mathit{then}\ \mathit{Min}\ \mathit{else}\ \mathit{Max}$ and f_R is defined similarly):

$$
\begin{array}{ll}
 & T(\mathcal{M}[\![Q \,[]\, R]\!]) \\
= & \{\ \text{by the argument just made}\ \} \\
 & \quad T(\Psi_Q) \setminus \{\, (e,t)\sigma \mid \exists\, e' \in E_R : l_R(e') = \mathbf{i} \wedge e \rightsquigarrow e' \wedge t > f_R(\mathcal{A}_R(e'))\,\} \\
 & \cup\, T(\Psi_R) \setminus \{\, (e,t)\sigma \mid \exists\, e' \in E_Q : l_Q(e') = \mathbf{i} \wedge e \rightsquigarrow e' \wedge t > f_Q(\mathcal{A}_Q(e'))\,\} \\
= & \{\ \text{manipulation}\ \} \\
 & \quad \{\, (e,t)\sigma \in T(\Psi_Q) \mid \neg(\exists\, e' \in E_R : l_R(e') = \mathbf{i} \wedge e \rightsquigarrow e' \wedge t > f_R(\mathcal{A}_R(e')))\} \\
 & \cup \{\, (e,t)\sigma \in T(\Psi_R) \mid \neg(\exists\, e' \in E_Q : l_Q(e') = \mathbf{i} \wedge e \rightsquigarrow e' \wedge t > f_Q(\mathcal{A}_Q(e')))\} \ \cup\ \{\varepsilon\} \\
= & \{\ \text{Definition 4.6}\ \} \\
 & \quad \{\, (e,t)\sigma \in T(\Psi_Q) \mid \neg(\exists\, e' \in E_R : l_R(e') = \mathbf{i} \ \wedge\ e \in \mathit{init}(\Psi_Q) \wedge \\
 & \qquad\qquad e' \in \mathit{init}(\Psi_R) \ \wedge\ t > f_R(\mathcal{A}_R(e')))\,\} \\
 & \cup\, \{(e,t)\sigma \in T(\Psi_R) \mid \neg(\exists\, e' \in E_Q : l_Q(e') = \mathbf{i} \ \wedge\ e \in \mathit{init}(\Psi_R) \wedge \\
 & \qquad\qquad e' \in \mathit{init}(\Psi_Q) \ \wedge\ t > f_Q(\mathcal{A}_Q(e')))\,\} \ \cup\ \{\varepsilon\} \\
= & \{\ \text{distributivity of } \wedge \text{ over } \vee\ ;\ e \in \mathit{init}(\Psi_Q) \text{ not bound in } \exists\ \} \\
 & \quad \{\, (e,t)\sigma \mid ((e,t)\sigma \in T(\Psi_Q) \ \wedge\ e \notin \mathit{init}(\Psi_Q)) \ \vee\ ((e,t)\sigma \in T(\Psi_Q) \wedge \\
 & \qquad \neg(\exists\, e' \in E_R : l_R(e') = \mathbf{i} \ \wedge\ e' \in \mathit{init}(\Psi_R) \ \wedge\ t > f_R(\mathcal{A}_R(e')))\,\} \\
 & \cup\, \{\, (e,t)\sigma \mid ((e,t)\sigma \in T(\Psi_R) \ \wedge\ e \notin \mathit{init}(\Psi_R)) \ \vee\ ((e,t)\sigma \in T(\Psi_R) \wedge \\
 & \qquad \neg(\exists\, e' \in E_Q : l_Q(e') = \mathbf{i} \ \wedge\ e' \in \mathit{init}(\Psi_Q) \ \wedge\ t > f_Q(\mathcal{A}_Q(e')))\,\} \cup \{\varepsilon\} \\
= & \{\ (e,t)\sigma \in T(\Psi_i) \wedge e \notin \mathit{init}(\Psi_i) \text{ is contradictory ; false} \vee P \equiv P\ \} \\
 & \quad \{\, (e,t)\sigma \in T(\Psi_Q) \mid \neg(\exists\, e' \in E_R : l_R(e') = \mathbf{i} \ \wedge\ e' \in \mathit{init}(\Psi_R) \ \wedge\ t > f_R(\mathcal{A}_R(e')))\,\} \\
 & \cup\, \{\, (e,t)\sigma \in T(\Psi_R) \mid \neg(\exists\, e' \in E_Q : l_Q(e') = \mathbf{i} \ \wedge\ e' \in \mathit{init}(\Psi_Q) \ \wedge\ t > f_Q(\mathcal{A}_Q(e')))\,\} \\
 & \cup \{\varepsilon\} \\
= & \{\ \text{Lemma 6.20 ; Lemma 6.21}\ \} \\
 & \quad \{\, (e,t)\sigma \in T(\Psi_Q) \mid R \overset{t}{\rightsquigarrow}\,\} \cup \{\, (e,t)\sigma \in T(\Psi_R) \mid Q \overset{t}{\rightsquigarrow}\,\} \cup \{\varepsilon\} \\
= & \{\ \text{Lemma 6.12}\ \} \\
 & \quad \{\, (e,t)\sigma \in T(\Psi_Q) \mid t \leqslant \mathsf{mi}(R)\,\} \cup \{\, (e,t)\sigma \in T(\Psi_R) \mid t \leqslant \mathsf{mi}(Q)\,\} \cup \{\varepsilon\} \\
= & \{\ \text{inductive hypothesis}\ \} \\
 & \quad \{\, (e,t)\sigma \in \mathcal{T}[\![Q]\!] \mid t \leqslant \mathsf{mi}(R)\} \cup \{\, (e,t)\sigma \in \mathcal{T}[\![R]\!] \mid t \leqslant \mathsf{mi}(Q)\} \cup \{\varepsilon\} \\
= & \{\ \text{Definition 6.18}\ \} \\
 & \quad \mathcal{T}[\![Q \,[]\, R]\!]
\end{array}
$$

4. *Relabelling.* Straightforward.

5. *Hiding.* Let $P = \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q$. The effect of $\mathcal{M}[\![\, \mathbf{hide}\ \Gamma\ \mathbf{in}\ Q\,]\!]$ is as follows: if $\sigma \in T(\Psi_Q)$ then (modulo turning some labels into i actions) $\sigma \in T(\Psi)$ unless:

   (a) some event, e_i say, in σ becomes newly immediate, i.e. $l_Q(e_i) \in \Gamma$, and $t_i > \mathit{Min}\,(\mathcal{Z}_Q(\sigma_i, e_i))$; or

   (b) there exists a point in the trace, e_i say, at which an event is enabled, that newly becomes immediate and must occur before t_i, i.e. $\exists\, e_i\, (\exists\, e \in \mathsf{en}_Q([\sigma_i]) : l_Q(e) \in \Gamma \ \wedge\ t_i > \mathit{Min}\,(\mathcal{Z}_Q(\sigma_i, e)))$.

However, since $e_i \in \mathsf{en}_Q([\sigma_i])$ we can see that the second of these conditions embraces the first. From this we can derive the following:

$$
\begin{array}{ll}
& T(\mathcal{M}[\![\,\mathbf{hide}\,\Gamma\ \mathbf{in}\ Q\,]\!]) \\
= & \quad \{\ \text{by the argument just made}\ \} \\
& \quad \{\,\sigma \setminus \Gamma \mid \sigma \in T(\Psi_Q) \ \wedge\ \neg(\exists\, e_i \in \overline{[\sigma]} : \exists\, e \in \mathsf{en}_Q([\sigma_i]) : (l_Q(e) \in \Gamma \ \wedge\ t_i{>}\mathit{Min}\,(\mathcal{Z}_Q(\sigma_i, e))))\} \\
= & \quad \{\ \text{manipulation}\ \} \\
& \quad \{\,\sigma \setminus \Gamma \mid \sigma \in T(\Psi_Q) \ \wedge\ \forall\, e_i \in \overline{[\sigma]} : \neg(\exists\, e \in \mathsf{en}_Q([\sigma_i]) : (l_Q(e) \in \Gamma \ \wedge\ t_i{>}\mathit{Min}\,(\mathcal{Z}_Q(\sigma_i, e))))\} \\
= & \quad \{\ \text{Lemma 6.22}\ \} \\
& \quad \{\,\sigma \setminus \Gamma \mid \sigma \in T(\Psi_Q)\ \wedge\ \forall\, i : (\neg(t_i{-}t_{i-1} > \mathsf{ma}(\Gamma, \mathsf{aft}(Q, \sigma_i))))\} \\
= & \quad \{\ \text{inductive hypothesis}\ \} \\
& \quad \{\,\sigma \setminus \Gamma \mid \sigma \in \mathcal{T}[\![\,Q\,]\!]\ \wedge\ \forall\, i : (t_i{-}t_{i-1} \leqslant \mathsf{ma}(\Gamma, \mathsf{aft}(Q, \sigma_i)))\} \\
= & \quad \{\ \text{Definition 6.18}\ \} \\
& \quad \mathcal{T}[\![\,\mathbf{hide}\,\Gamma\ \mathbf{in}\ Q\,]\!]
\end{array}
$$

6. *Enabling.* Let $P = Q >\!\!> R$. Timed traces, σ say, of Ψ arise for one of the following reasons:

   (a) σ is a timed trace of Ψ_Q, it does not end with a δ and no event in the trace is prevented from happening by virtue of being pre-empted by a newly urgent event, i.e. a successful termination event. To be more precise:

   $$\sigma \in T(\Psi_Q)\ \wedge\ \sigma \neq \sigma'(e_\delta, t)\ \wedge \\ \neg(\exists\, i : \exists\, e \in \mathsf{en}_Q([\sigma_i]) : (l_Q(e) = \delta\ \wedge\ t_i > \mathit{Min}\,(\mathcal{Z}_Q(\sigma_i, e))))$$

   (b) $\sigma = \sigma_1(e_{\mathbf{i}}, t)^t[\sigma_2]$ where $\sigma_1(e_\delta, t) \in T(\Psi_Q)$ and $\sigma_2 \in T(\Psi_R)$ and no event in $\sigma_1(e_\delta, t)$ is prevented from happening because it can be pre-empted by a newly urgent event, i.e. a successful termination event. This embraces the possibility that $e_{\mathbf{i}}$ cannot happen at a certain time because it would become urgent earlier. To be more precise:

   $$\sigma = \sigma_1(e_{\mathbf{i}}, t)^t[\sigma_2]\ \wedge\ \sigma_1(e_\delta, t) \in T(\Psi_Q)\ \wedge\ \sigma_2 \in T(\Psi_R) \wedge \\ \neg(\exists\, i : \exists\, e \in \mathsf{en}_Q([(\sigma_1)_i]) : (l_Q(e) = \delta\ \wedge\ t_i > \mathit{Min}(\mathcal{Z}_Q((\sigma_1))_i, e))))$$

On the basis of this discussion we can derive the following:

$$
\begin{array}{ll}
& T(\mathcal{M}[\![\,Q >\!\!> R\,]\!]) \\
= & \quad \{\ \text{from above discussion}\ \} \\
& \quad \{\,\sigma \in T(\Psi_Q) \mid \sigma \neq \sigma'(e_\delta, t) \ \wedge \\
& \qquad\qquad \forall\, i : \neg(\exists\, e \in \mathsf{en}_Q([\sigma_i]) : (l_Q(e) = \delta\ \wedge\ t_i > \mathit{Min}\,(\mathcal{Z}_Q(\sigma_i, e))))\} \\
& \quad \cup\, \{\,\sigma_1(e_{\mathbf{i}}, t)^t[\sigma_2] \mid \sigma_1(e_\delta, t) \in T(\Psi_Q)\ \wedge\ \sigma_2 \in T(\Psi_R) \ \wedge \\
& \qquad\qquad \forall\, i : \neg(\exists\, e \in \mathsf{en}_Q([(\sigma_1)_i]) : (l_Q(e) = \delta\ \wedge\ t_i > \mathit{Min}(\mathcal{Z}_Q((\sigma_1))_i, e))))\} \\
= & \quad \{\ \text{Lemma 6.22}\ \} \\
& \quad \{\,\sigma \in T(\Psi_Q) \mid \sigma \neq \sigma'(e_\delta, t)\ \wedge\ \forall\, i : \neg(t_i{-}t_{i-1} > \mathsf{ma}(\{\delta\}, \mathsf{aft}(Q, \sigma_i)))\,\} \\
& \quad \cup\, \{\,\sigma_1(e_{\mathbf{i}}, t)^t[\sigma_2] \mid \sigma_1(e_\delta, t) \in T(\Psi_Q)\ \wedge\ \sigma_2 \in T(\Psi_R) \ \wedge \\
& \qquad\qquad \forall\, i : \neg(t_i{-}t_{i-1} > \mathsf{ma}(\{\delta\}, \mathsf{aft}(Q, (\sigma_1))_i)))\,\} \\
= & \quad \{\ \text{Inductive hypothesis ; manipulation}\ \} \\
& \quad \{\,\sigma \in \mathcal{T}[\![\,Q\,]\!] \mid \sigma \neq \sigma'(e_\delta, t)\ \wedge\ \forall\, i : t_i - t_{i-1} \leqslant \mathsf{ma}(\{\delta\}, \mathsf{aft}(Q, \sigma_i))\,\} \\
& \quad \cup\, \{\,\sigma_1(e_{\mathbf{i}}, t)^t[\sigma_2] \mid \sigma_1(e_\delta, t) \in \mathcal{T}[\![\,Q\,]\!]\ \wedge\ \sigma_2 \in \mathcal{T}[\![\,R\,]\!] \ \wedge \\
& \qquad\qquad \forall\, i : t_i{-}t_{i-1} \leqslant \mathsf{ma}(\{\delta\}, \mathsf{aft}(Q, (\sigma_1)_i))\,\} \\
= & \quad \{\ \text{Definition 6.18}\ \} \\
& \quad \mathcal{T}[\![\,Q >\!\!> R\,]\!]
\end{array}
$$

7. *Disrupt.* Let $P = Q \,[\!>\, R$. The effect of $Q \,[\!>\, R$ is to union the components of $\mathcal{M}[\![Q]\!]$ and $\mathcal{M}[\![R]\!]$ apart from conflict. Here, in addition to unioning the existing conflicts, new asymmetric conflicts from the events of Q to the initial events of R and from the initial events of R to the exit events of Q are included. So, for σ to be in $T(\Psi)$ one of the following must hold (in the following $f = \mathit{if}\ e \in \mathcal{I}\ \mathit{then}\ \mathit{Min}\ \mathit{else}\ \mathit{Max}$ and $f_j = \mathit{if}\ e \in \mathcal{I}_j\ \mathit{then}\ \mathit{Min}\ \mathit{else}\ \mathit{Max}$ for $j \in \{Q, R\}$):

   (a) $\sigma \in T(\Psi_Q)$ and the new conflicts created in $Q \,[\!>\, R$ do not cause condition 4 of Definition 3.8 to fail (the other conditions will hold automatically). Thus, we require the following:

$$\sigma \in T(\Psi_Q) \ \wedge\ \forall i : (\forall\, e \in \mathsf{en}([\sigma_i]) : l(e) = \mathbf{i} \implies t_i \leqslant f(\mathcal{Z}(\sigma_i, e)))$$

   In fact, by the nature of disrupt, a trace of Ψ_Q will only fail to be a trace of Ψ if the last element of σ is prevented by a newly in conflict urgent event. Thus, we can rewrite this condition to,

$$\sigma = \sigma'(e,t) \in T(\Psi_Q) \ \wedge\ (\forall\, e' \in \mathsf{en}([\sigma']) : l(e') = \mathbf{i} \implies t \leqslant f(\mathcal{Z}(\sigma', e'))) \quad (*)$$

   Furthermore, since $\sigma'(e,t) \in T(\Psi_Q)$ we know that $\forall\, e' \in \mathsf{en}([\sigma']) \cap E_Q$ the property, $l(e') = \mathbf{i} \implies t \leqslant f(\mathcal{Z}(\sigma', e))$ holds. In addition, since σ' does not contain any events from E_R and labelling is preserved by disruption, we can see that condition (*) is equivalent to the following:

$$\sigma = \sigma'(e,t) \in T(\Psi_Q) \ \wedge\ (\forall\, e' \in \mathsf{en}_R([\varepsilon]) : l_R(e') = \mathbf{i} \implies t \leqslant f_R(\mathcal{Z}_R(\varepsilon, e')))$$

   where the subscripts here refer to the relevant time-extended event structure, i.e. Ψ_R.

   (b) $\sigma \in T(\Psi_R)$ and the new conflicts created in $Q \,[\!>\, R$ do not cause condition 4 of Definition 3.8 to fail (once again the other conditions hold automatically). Thus, we require the following:

$$\sigma \in T(\Psi_R) \ \wedge\ \forall i : (\forall\, e \in \mathsf{en}([\sigma_i]) : l(e) = \mathbf{i} \implies t_i \leqslant f(\mathcal{Z}(\sigma_i, e)))$$

   In fact, by the nature of disrupt, a trace of Ψ_Q will only fail to be a trace of Ψ if the first element of σ is prevented by a newly in conflict urgent event. Thus, we can rewrite the above condition to:

$$\sigma = (e,t)\sigma' \in T(\Psi_R) \ \wedge\ (\forall\, e' \in \mathsf{en}([\varepsilon]) : l(e') = \mathbf{i} \implies t \leqslant f(\mathcal{Z}(\varepsilon, e')))$$

   In addition, by observing that $(e,t)\sigma' \in T(\Psi_R)$ we can, in a similar manner to that employed to reduce (*), see that the following condition is equivalent:

$$\sigma = (e,t)\sigma' \in T(\Psi_R) \ \wedge\ (\forall\, e' \in \mathsf{en}_Q([\varepsilon]) : l_Q(e') = \mathbf{i} \implies t \leqslant f_Q(\mathcal{Z}_Q(\varepsilon, e')))$$

   (c) $\sigma = \sigma_1\sigma_2$ where $\sigma_1 \in T(\Psi_Q)$ and $\sigma_2 \in T(\Psi_R)$, i.e. the concatenation of traces from Q and R. However, certain additional conditions have to be up held.

      i. σ_1 cannot contain an exit event.
      ii. The last element of σ_1 cannot be prevented by a newly in conflict urgent event.
      iii. The first element of σ_2 cannot be prevented by a newly in conflict urgent event.
      iv. The concatenation of σ_1 and σ_2 is time consistent.

   To formalise these properties we get:

$$\begin{array}{l}\sigma = \sigma_1\sigma_2 \wedge \sigma_1 = \sigma'(e,t) \in T(\Psi_Q) \wedge \sigma_2 = (e',t')\sigma'' \in T(\Psi_R) \wedge l(e) \neq \delta \wedge t \leqslant t' \\ (\forall\, e \in \mathsf{en}([\sigma']) : l(e) = \mathbf{i} \Rightarrow t \leqslant f(\mathcal{Z}(\sigma', e))) \wedge \\ (\forall\, e \in \mathsf{en}([\sigma_1]) : l(e) = \mathbf{i} \Rightarrow t' \leqslant f(\mathcal{Z}(\sigma_1, e))) \wedge\end{array}$$

   and once again we can reduce these properties in a similar manner to that used to reduce (*) above.

$$\begin{array}{l}
\sigma = \sigma_1\sigma_2 \ \wedge \ \sigma_1 = \sigma'(e,t) \in T(\Psi_Q) \ \wedge \\
\{\, \sigma(e,t) \in T(\Psi_Q) \mid \sigma_2 = (e',t')\sigma'' \in T(\Psi_R) \ \wedge \ l(e) \neq \delta \ \wedge \ t \leqslant t' \\
(\forall\, e \in \mathsf{en}_R([\varepsilon]) : l_R(e) = \mathbf{i} \Rightarrow t \leqslant f_R(\mathcal{Z}_R(\varepsilon, e))) \ \wedge \\
(\forall\, e \in \mathsf{en}_Q[\sigma_1]) : l_Q(e) = \mathbf{i} \Rightarrow t' \leqslant f_Q(\mathcal{Z}_Q(\sigma_1, e))) \ \wedge
\end{array}$$

Now we can put these conditions together to obtain the following:

$$\begin{array}{ll}
& T(\mathcal{M}[\![Q \,[>\, R]\!]) \\
= & \quad \{ \text{ from above discussion } \} \\
& \quad \{\, \sigma(e,t) \in T(\Psi_Q) \mid \forall\, e' \in \mathsf{en}_R([\varepsilon]) : l_R(e') = \mathbf{i} \ \Rightarrow \ t \leqslant f_R(\mathcal{Z}_R(\varepsilon, e')) \,\} \\
& \cup \{\, (e,t)\sigma \in T(\Psi_R) \mid \forall\, e' \in \mathsf{en}_Q([\varepsilon]) : l_Q(e') = \mathbf{i} \ \Rightarrow \ t \leqslant f_Q(\mathcal{Z}_Q(\varepsilon, e')) \,\} \\
& \cup \{\, \sigma_1\sigma_2 \mid \sigma_1 = \sigma'(e,t) \in T(\Psi_Q) \ \wedge \ \sigma_2 = (e',t')\sigma'' \in T(\Psi_R) \ \wedge \\
& \qquad\qquad l(e) \neq \delta \ \wedge \ (\forall\, e \in \mathsf{en}_R([\varepsilon]) : l_R(e) = \mathbf{i} \Rightarrow t \leqslant f_R(\mathcal{Z}_R(\varepsilon, e))) \ \wedge \\
& \qquad\qquad (\forall\, e \in \mathsf{en}_Q([\sigma_1]) : l_Q(e) = \mathbf{i} \Rightarrow t' \leqslant f_Q(\mathcal{Z}_Q(\sigma_1, e))) \ \wedge \ t \leqslant t' \,\} \\
= & \quad \{ \text{ since the labelling of } E_Q \text{ is preserved ; Lemma 6.21 } \} \\
& \quad \{\, \sigma(e,t) \in T(\Psi_Q) \mid R \overset{t}{\rightsquigarrow} \,\} \cup \{\, (e,t)\sigma \in T(\Psi_R) \mid Q \overset{t}{\rightsquigarrow} \,\} \\
& \cup \{\, \sigma_1\sigma_2 \mid \sigma_1 = \sigma'(e,t) \in T(\Psi_Q) \ \wedge \ \sigma_2 = (e',t')\sigma'' \in T(\Psi_R) \ \wedge \\
& \qquad\qquad e \neq e_\delta \ \wedge \ R \overset{t}{\rightsquigarrow} \ \wedge\, \mathsf{aft}(Q, \sigma_1) \overset{t'-t}{\rightsquigarrow} \ \wedge\, t \leqslant t' \,\} \\
= & \quad \{ \text{ Lemma 6.12 ; inductive hypothesis } \} \\
& \quad \{\, \sigma(e,t) \in \mathcal{T}[\![Q]\!] \mid t \leqslant \mathsf{mi}(R) \,\} \cup \{\, (e,t)\sigma \in \mathcal{T}[\![R]\!] \mid t \leqslant \mathsf{mi}(Q) \,\} \\
& \cup \{\, \sigma_1\sigma_2 \mid \sigma_1 = \sigma'(e,t) \in \mathcal{T}[\![Q]\!] \ \wedge \ \sigma_2 = (e',t')\sigma'' \in \mathcal{T}[\![R]\!] \ \wedge \\
& \qquad\qquad e \neq e_\delta \ \wedge \ t \leqslant \mathsf{mi}(R) \ \wedge \ 0 \leqslant t'-t \leqslant \mathsf{mi}(\mathsf{aft}(Q, \sigma_1)) \,\} \\
= & \quad \{ \text{ Definition 6.18 } \} \\
& \quad \mathcal{T}[\![Q \,[>\, R]\!]
\end{array}$$

8. *Parallel Composition.* Since in our model only internally labelled events are urgent and such events cannot be synchronised on it is easy to verify that no new asymmetric conflicts are introduced between urgent events in E_Q and in E_R. Consequently, $\sigma \in T(\mathcal{M}[\![Q \,|[\Gamma]|\, R]\!])$ if and only if $\pi_Q(\sigma) \in T(\mathcal{M}[\![Q]\!])$, $\pi_R(\sigma) \in T(\mathcal{M}[\![R]\!])$ and σ is time consistent. So, we have the following:

$$\begin{array}{ll}
& T(\mathcal{M}[\![Q \,|[\Gamma]|\, R]\!]) \\
= & \quad \{ \text{ by above discussion } \} \\
& \quad \{\, \sigma \in (\overline{T(\Psi_Q)} \ \bowtie_\Gamma \ \overline{T(\Psi_R)})^* \mid \pi_Q(\sigma) \in T(\Psi_Q) \ \wedge \ \pi_R(\sigma) \in T(\Psi_R) \,\} \\
= & \quad \{ \text{ inductive hypothesis } \} \\
& \quad \{\, \sigma \in (\overline{\mathcal{T}[\![Q]\!]} \ \bowtie_\Gamma \ \overline{\mathcal{T}[\![R]\!]})^* \mid \pi_Q(\sigma) \in \mathcal{T}[\![Q]\!] \ \wedge \ \pi_R(\sigma) \in \mathcal{T}[\![R]\!] \,\} \\
= & \quad \{ \text{ Definition 6.18 } \} \\
& \quad \mathcal{T}[\![Q \,|[\Gamma]|\, R]\!]
\end{array}$$

(End of proof.)

6.5 Recursive behaviours

In the previous section we considered the consistency for non-recursive processes. This section is devoted to extending the consistency results to recursive processes. We assume in this section that all process instantiations are guarded, i.e., all process instantiations are preceded by an action-prefix or an enable operator.

Definition 6.24 (Substitution of process instantiations)
For process instantiation x, $Q[x := P]$, is defined as

1. $\mathbf{stop}[x := P] \triangleq \mathbf{stop}$
2. $\mathbf{exit}\,\{\,T\,\}[x := P] \triangleq \mathbf{exit}\,\{\,T\,\}$
3. $(\mathsf{op}\,Q)[x := P] \triangleq \mathsf{op}\,(Q[x := P])$ for $\mathsf{op} \in \{\, a\,\{\,T\,\}\;;\;, \mathbf{hide}, [\,], \mathbf{Wait}\,\}$
4. $(Q\;\mathsf{op}\,Q')[x := P] \triangleq (Q[x := P])\,\mathsf{op}\,(Q'[P := P])$ for $\mathsf{op} \in \{\;[\,]\;, >>, [>, |[]|\;\}$
5. $Q[x := P] \triangleq \begin{cases} \pi(x) & \text{if } Q = x_\pi \\ Q & \text{if } Q \neq x. \end{cases}$

□

$Q[x := P]$ denotes behaviour Q where all occurrences of x_π in Q are replaced with $\pi(P)$, the process body of x with renaming $\pi()$ applied. As a next subsidiary notion we define the approximations of x. The n-th approximation of x is defined as the n-th unfolding where each occurrence of x is replaced by **stop**.

Definition 6.25 (Approximations)
For $x := Q$ the n-th *approximation* of x, denoted x^n, is defined as: $x^0 \triangleq \mathbf{stop}$ and $x^{n+1} \triangleq Q[x := x^n]$. □

Theorem 6.26 (Traces of a process instantiation)
For process instantiation x we have $\mathcal{T}[\![\,x\,]\!] = \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}[\![\,x^j\,]\!]$.

Proof:

Analogous to the proof in KATOEN ET. AL [19]. *(End of proof.)*

Due to this characterisation of the traces of recursive behaviour x in terms of approximations of x (which are *finite* expressions) it is clear that Lemma 6.19 holds for all expressions, including the recursive ones.

For the result we have been working up to we need an additional lemma:

Lemma 6.27
For $x := Q$ we have: $\forall\, i \geqslant 0 : \mathcal{M}[\![\,x^i\,]\!] = \mathcal{F}^i_Q(-)$.

Proof:

By induction on i. Straightforward and omitted. *(End of proof.)*

Theorem 6.28 (Compatibility theorem)
For guarded recursive ET-LOTOS expression P: $T(\mathcal{M}[\![\,P\,]\!]) = \mathcal{T}[\![\,P\,]\!]$.

Proof:

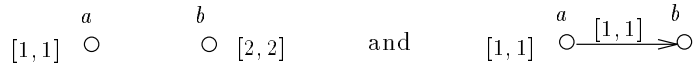
For finite behaviours P the theorem follows directly from Lemmas 6.23 and 6.19. For guarded recursive behaviours the proof is as follows. Consider $x := Q$. Then we derive:

$$
\begin{array}{ll}
 & T(\mathcal{M}[\![\,x\,]\!]) \\
= & \{\ \text{denotational semantics of } x\ \} \\
 & T(\bigsqcup_i \mathcal{F}^i_Q(-)) \\
= & \{\ \text{Lemma 6.27}\ \} \\
 & T(\bigsqcup_i \mathcal{M}[\![\,x^i\,]\!]) \\
= & \{\ \text{Theorem 4.33}\ \}
\end{array}
$$

$$
\begin{array}{ll}
 & \bigcup_i \bigcap_{j \geqslant i} T(\mathcal{M}[\![x^j]\!]) \\
= & \quad \{ \text{ Lemmata 6.23 and 6.19 } \} \\
 & \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}[\![x^j]\!] \\
= & \quad \{ \text{ Theorem 6.26 } \} \\
 & \mathcal{T}[\![x]\!]
\end{array}
$$

(End of proof.)

Theorem 6.28 extends the consistency result for the untimed case by Langerak. For the untimed case it can be proven that two event structures that have identical event traces also have the same (set of) partial orders. So, event trace equivalence implies (in fact, is equivalent to) partial-order equivalence. For our timed case this is not the case. For instance,



are timed event-trace equivalent, but are different from a partial order perspective: in the first structure e_a and e_b are independent, whereas in the second they are not. It turns out that timed event-trace equivalence and partial-order equivalence coincide when resorting to "ill-timed" traces, traces that do respect causalities but do not necessarily advance in time [1]. Then the above two above event structures are distinguished, since $(e_b, 2)(e_a, 1)$ is an ill-timed trace of the left, but not of the right structure. This is possible for cases where urgency is absent, or where urgency is restricted for the sole purpose of modelling timeouts [21]. Since urgency does play such a prominent role in TE-LOTOS it relates causally independent events.

7 Related Work

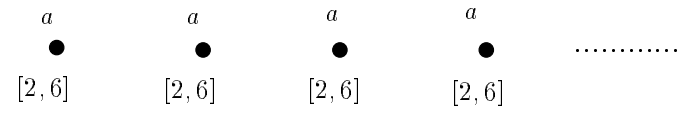
7.1 Alternative semantics for TE-LOTOS

The semantics of the timed component of enhanced LOTOS which is currently proposed for standardisation is based on the operational interleaving semantics of Léonard & Leduc [26, 27]. Basically this semantics can be obtained from our event-based operational semantics when omitting the event identifiers. (The only minotr technical difference is that we use predicates rather than negative premises; see the discussion on this issue at the end of Section 5). The timed event structure semantics presented in this report is consistent with this operational semantics for guarded recursive processes. This means that for guarded recursive processes the 'interleaved' view of the event structure semantics and the operational interleaving semantics of TE-LOTOS in [26, 27] correspond.

Unguarded recursion behaves differently in the event structure setting, to how it behaves in the interleaved setting. Consider, for instance:

$$\mathsf{Unguarded} := a\,\{\,2..6\,\}\,;\,\mathbf{stop} \,|||\, \mathsf{Unguarded}$$

The interleaving semantics of TE-LOTOS will generate a time blockage for this behaviour, i.e. instantiation of Unguarded in any context will block the passage of time and the whole system (!) is not be able to progress. In contrast, the event structure semantics for a process instantiation of Unguarded,

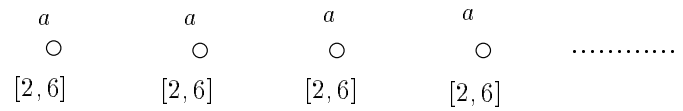


does not cause a time-blocking behaviour. This event structure allows (amongst others) a trace of infinite length consisting of events all labelled with a that occur in the interval $[2,6]$. For $\mathsf{Unguarded} \,|||\, b\,(t)\,;\,\mathbf{stop}$ for arbitrary t ($t \neq \infty$) this means that the occurrence of b is not avoided: b can happen after any finite sequence of a's. In the interleaving semantics b is permanently disabled.

Remark the difference with the process

$$\mathsf{Unguarded}' := \mathbf{hide}\ a\ \mathbf{in}\ \mathsf{Unguarded}$$

which leads to the time-extended event structure

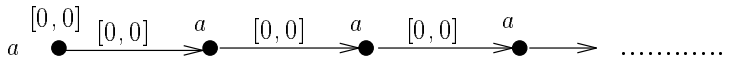


This event structure has the same traces as $\mathcal{M}[\![\,\mathsf{Unguarded}\,]\!]$, but $\mathsf{Unguarded}' \,|||\, b\,(t)\,;\,\mathbf{stop}$ now only permits b to happen if $t \leqslant 6$ since at time 6 the infinite amount of internal urgent events is forced to happen. For $t > 6$ there exist an infinite number of urgent events that should occur before b. This effect is due to the interplay between urgency and unguarded recursion and would, for instance, be avoided by requiring time-guardedness.

Bryans, Davies & Schneider thoroughly investigated a denotational semantics for TE-LOTOS based on a timed failures model. In [11] they present a metric denotational semantics which is rather simple, but which does not support non-time-guarded (or, instant) recursive processes, that is, processes in which a recursive call is possible without delay. [12] presents a denotational semantics based on a cpo approach that is able to cope with instant recursive processes. A more involved notion of timed failure is needed in this case to keep track of the time at which an infinite sequence of internal actions, i.e., a divergence, has taken place. Remark that instant recursive processes do not pose any problem in our approach. For instance process:

$$\mathsf{Instantrec} := a\,(0)\,;\,\mathsf{Instantrec}$$

will give rise to the time-extended event structure

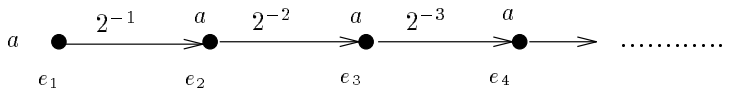


This structure can perform infinitely many events labelled a without spending time. (Although this process is not implementable such processes are useful, as also observed in [12], in a constraint-oriented specification style [36].)

In the timed-failures setting of [12] two other cases led to a refinement of the notion of timed failure: Zeno processes and timestops introduced by the use of selection predicates. A typical Zeno process is

$$\begin{aligned} \mathsf{Zeno} &::= \mathsf{Zeno}_1 \\ \mathsf{Zeno}_k &::= a\,;\,\mathbf{Wait}(2^{-k})\,;\,\mathsf{Zeno}_{k+1} \text{ for } k \geqslant 1 \end{aligned}$$

This process is ready to perform an a, delays for 0.5 time units, is then available for performing another a, then waits 0.25 time units, and so on, ad infinitum. The timed non-interleaving semantics for process Zeno is



The infinite sequence $(e_1, t_1)(e_2, t_2)\ldots$ is an event trace of this timed event structure if $t_{i+1} \geqslant t_i + 2^{-i}$ for all $i \geqslant 1$. In particular, for $t_1 = 0$ and $t_{i+1} = t_i + 2^{-i}$ we obtain an event trace in which infinitely many events happen before time 1.

The last example concerns the use of selection predicates. Consider:

$$\mathsf{Time} := \textbf{hide } a \textbf{ in } (a\,(t)[t > 1]\ ;\ \textbf{stop})$$

The selection predicate $[t > 1]$ insists that a may be performed at any time beyond 1, excluding time 1 itself. Since a is hidden it should occur as soon as possible. But, since there is no earliest time after 1, it means that this process is permanently disabled, and more importantly, in the interleaving semantics of Léonard & Leduc [26] gives rise to a (global) time block. Although we do not deal with data (and selection predicates) in this report, the natural event structure corresponding to process Time would be

This event structure only has the empty event trace.

7.2 Related timed partial-order models

Various timed extensions of partial-order models, such as configurations, Mazurkiewicz traces, higher-order automata, pomsets, and event automata are known from the literature, but these models are (to the knowledge of the authors) not used as semantical models for timed process algebras. An exception is the work of Fidge [14] who uses a timed extension of causal trees to provide a semantics for a timed version of CCS. Causal trees are event structures modulo history-preserving bisimulation. Each event in a causal tree has a set of backward pointers to each event on which it causally depends. In summary, the approach of Fidge [14] is as follows. Time constraints are expressed by associating a set of relative times to events. The relative delays T state that an event can only occur at t time units (for some $t \in T$) 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 real-time semantics of CCS is defined operationally. Due to the adjustments of backward pointers the inference rules are somewhat complicated.

8 Concluding Remarks

In this report we presented a non-interleaving semantics for ET-LOTOS, the timed process algebraic part on which the timed component of the enhanced LOTOS language which is currently under standardisation is based. The semantics is based on a suitable timed extension of extended bundle event structures, a model developed by Langerak [23, 24] for defining a branching-time non-interleaving semantics for LOTOS.

The timed extension of event structures introduced in this report is tailored to the features of ET-LOTOS, in particular to the combination of urgency and maximal progress in that

language. In this way this work differs from earlier work on timed extensions of bundle event structures in which the authors were involved [5, 8, 9, 21, 19].

Our non-interleaving semantics complements the existing operational interleaving semantics of ET-LOTOS [26, 27]. We showed that for guarded recursive processes there exists a strong consistency relation between the denotational event structure semantics and an event-based operational semantics, from which the interleaving semantics of ET-LOTOS can easily be obtained by omitting the event identifiers. By defining the operational semantics using an auxiliary function defined by induction on the structure of expressions, the use of negative premises could be avoided.

In our opinion, one of the nice features of the timed extension of event structures is the absence of an explicit mechanism to advance time. As in physics, time flows implicitly and is treated in the model just as a parameter. An important consequence of this feature is that unguarded recursive processes, instant recursive processes (i.e., processes that take no time until the next recursive call) and Zeno processes can be treated in a rather perspicuous way. Technically speaking, this means in principal that such processes do not avoid the passage of time and, so, do not affect the evolvement of other (possibly independent) processes. [5] This contrasts with the interleaving semantics of ET-LOTOS (and various other timed process algebras) that force processes to synchronise on the passage of time such that a single process that blocks the advancement of time automatically results in a *global* blockage of time, thus preventing other processes from evolving.

It is for further work to incorporate data and to take the recent developments in the enhanced LOTOS language into account [17]. Since the timed component of the current enhanced LOTOS language is based on the same ideas that formed the basis of this report, we believe that our current model provides a good basis for these extensions.

Acknowledgements: The authors would like to thank Luc Léonard and Guy Leduc for discussions on the semantics of ET-LOTOS.

References

[1] L. Aceto and D. Murphy. Timing and causality in process algebra. *Acta Informatica*, **33**:317–350, 1996.

[2] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks & ISDN Systems*, **14**:25–59, 1987.

[3] T. Bolognesi, F. Lucidi and S. Trigila. Converging towards a timed LOTOS standard. *Computer Standards & Interfaces*, **16**:87–118, 1994.

[4] G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. *Information & Computation*, **114**:247–314, 1994.

[5] H. Bowman. A true concurrency approach to time extended LOTOS (revised version). Technical Report 17-96, University of Kent at Canterbury, 1996.

[5] An exception to this case is when an infinite number of urgent actions can be generated in a finite amount of time, as in

$$\mathsf{Infurgent} := \mathbf{hide}\ a\ \mathbf{in}\ a\ (0)\ ;\ \mathsf{Infurgent}$$

Process $\mathsf{Infurgent} \,|||\, b\ (1)\ ;\ \mathbf{stop}$ can now never perform a b action, since there always exists an urgent a action that should happen before time 1.

[6] H. Bowman, G.S. Blair, L. Blair and A.G. Chetwynd. Time versus abstraction in formal description. In R.L. Tenney, P.D. Amer, and M.Ü. Uyar, editors, *Formal Description Techniques VI*, volume C–22 of *IFIP Transactions*, pages 467–482. North-Holland, 1994.

[7] H. Bowman, L. Blair, G.S. Blair and A.G. Chetwynd. Formal description of distributed multimedia systems; an assessment of potential techniques. *Computer Communications*, **18**(12):964–977, 1995.

[8] H. Bowman and J. Derrick. Extending LOTOS with time; a true concurrency perspective. In M. Bertran and T. Rus, editors, *Proceedings 4th Amast Workshop on Real-Time Systems, Concurrent and Distributed Software*, LNCS 1231, pages 383–399. Springer-Verlag, 1997.

[9] 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*, pages 309–337. World Scientific, 1994.

[10] E. Brinksma, J-P. Katoen, R. Langerak and D. Latella. A stochastic causality-based process algebra. *The Computer Journal*, **38**(7):552–565, 1995.

[11] J.W. Bryans, J. Davies and S.A. Schneider. Towards a denotational semantics for ET-LOTOS. In I. Lee and S.A. Smolka, editors, *Concur '95*, LNCS 962, pages 269–283. Springer-Verlag 1995.

[12] J. Davies, J.W. Bryans and S.A. Schneider. Real-time LOTOS and timed observations. In D. Hogrefe and S. Leue, editors, *Formal Description Techniques VIII*. Chapmann & Hall, 1995.

[13] J. Engelfriet. Determinacy $\rightarrow$ (observation equivalence = trace equivalence). *Theoretical Computer Science*, **36**:21–25, 1985.

[14] C.J. Fidge. A constraint-oriented real-time process calculus. In M. Diaz and R. Groz, editors, *Formal Description Techniques V*, volume C–10 of *IFIP Transactions*, pages 363–378. North-Holland, 1993.

[15] J.F. Groote. Transition system specifications with negative premises. *Theoretical Computer Science*, **188**:263–299, 1993.

[16] ISO/IEC JTC1/SC21/WG1 N1173. Revised Working Draft on Enhancements to LOTOS. September 1996.

[17] ISO/IEC JTC1/SC21/WG. Working Draft on Enhancements to LOTOS. January 1997.

[18] J-P. Katoen. *Quantitative and Qualitative Extensions of Event Strcutures*. PhD thesis, University of Twente, 1996.

[19] J-P. Katoen, R. Langerak, E. Brinksma, D. Latella and T. Bolognesi. A consistent causality-based view on a timed process algebra including urgent interactions. *Journal of Formal Methods in System Design*, 1997. (extended abstract in A. Cornell and D. Ionescu, editors, *Proceedings 3rd Amast Workshop on Real-Time System Development*, pages 212–227, 1996.)

[20] 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*, pages 253–268. North-Holland, 1994.

[21] 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*, LNCS 1135, pages 385–405. Springer-Verlag, 1996.

[22] J-P. Katoen, D. Latella, R. Langerak and E. Brinksma. Stochastic simulation of event structures. In M. Ribaudo, editor, *Proceedings 4th Int. Workshop on Process Algebra and Performance Modelling*, pages 21–40. C.L.U.T. Press, 1996.

[23] R. Langerak. *Transformations and Semantics for LOTOS*. PhD thesis, University of Twente, 1992.

[24] R. Langerak. Bundle event structures: a non-interleaving semantics for LOTOS. In M. Diaz and R. Groz, editors, *Formal Description Techniques V*, volume C–10 of *IFIP Transactions*, pages 331–346. North-Holland, 1993.

[25] R. Langerak, E. Brinksma and J-P. Katoen. Causal ambiguity and partial orders in event structures. In A. Mazurkiewicz and J. Winkowski, *Proceedings of the 7th Int. Conference Concur'97: Concurrency Theory*. LNCS 1243, pages 317–332. Springer-Verlag 1997.

[26] L. Léonard and G. Leduc. A formal definition of time in LOTOS – extended version. Technical report, Université de Liège, 1995.

[27] L. Léonard and G. Leduc. An introduction to ET-LOTOS for the description of time-sensitive systems. *Computer Networks & ISDN Systems* **29**(3):271–292, 1997.

[28] R. Loogen and U. Goltz. Modelling nondeterministic concurrent processes with event structures. *Fundamenta Informaticae*, **14**:39–74, 1991.

[29] Z. Manna, S. Ness and J. Vuillemin. Inductive methods for proving properties of programs. *Communications of the ACM*, **16**(8):491–502, 1973.

[30] R. Milner. Process constructors and interpretations. In *Information Processing 86*. Elsevier, 1986.

[31] F. Moller and C. Tofts. A temporal calculus of communicating systems. In J.C.M. Baeten and J.W. Klop, editors, *Concur '90*, LNCS 458. Springer-Verlag, 1990.

[32] M. Nielsen, G.D. Plotkin and G. Winskel. Petri nets, event structures and domains, part 1. *Theoretical Computer Science*, **13**(1):85–108, 1981.

[33] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In J.W. de Bakker, C. Huizing, W.-P. de Roever, and G. Rozenberg, editors, *Real-Time: Theory in Practice*, LNCS 600, pages 526–548. Springer-Verlag, 1992.

[34] G.M. Pinna and A. Poigné. On the nature of events: another perspective in concurrency. *Theoretical Computer Science*, **138**(2):425–454, 1995.

[35] J. Stoy. *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. MIT Press, Cambridge, 1977.

[36] C.A. Vissers, G. Scollo, M. van Sinderen and E. Brinksma. On the use of specification styles in the design of distributed systems. *Theoretical Computer Science*, **89**(1):179–206, 1991.

[37] 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*, LNCS 354, pages 364–397. Springer-Verlag, 1989.