Towards weak sequencing for E-LOTOS

M. Kapus-Kolar

Jožef Stefan Institute, POB 3000, SI-1001 Ljubljana, Slovenia

Received 11 August 2004

Abstract

In E-LOTOS, a standard process-algebraic language for specification of concurrent and reactive real-time systems, the only form of process sequencing is strong sequencing, meaning that no action of a particular process is ever allowed to occur before complete termination of the preceding process. In the paper, we propose how to enhance the language with weak sequencing, facilitating specification of accelerated action execution, i.e. of partially overlapping processes. Defining an enhanced operational semantics, we formalize the approach for discrete-time basic E-LOTOS processes and give informal guidelines for its generalization to full E-LOTOS.

Key words: Concurrent systems, Formal specification techniques, Basic E-LOTOS, Weak sequencing

1. Introduction

When one specifies that a process B_2 may start only after completion of another process B_1, that might be for two reasons. If the intention is to secure that the actions of B_2 come strictly after the actions of B_1, such strong sequencing of B_1 and B_2 is definitely the right choice. However, if the only reason is that in B_2, there is an event E_2 causally related to an event E_1 in B_1 (e.g. because E_2 uses the data produced by E_1), it might be desirable to allow commutation (or even concurrent execution) of pairs of causally unrelated actions E_1' from B_1 and E_2' from B_2 [8,10].

In many cases, such weak sequencing of B_1 and B_2 is even the only acceptable solution, e.g. in a real-time system executing an algorithm requiring that a particular E_2 in B_2 is executed as soon as possible, i.e. without waiting for completion of B_1. Another typical example where commutation of an E_1 in B_1 and an E_2 in B_2 is desirable is the case where E_1 and E_2 belong to concurrent components of a distributed system, so that they cannot be sequenced without additional time constraints or events for inter-component communication. Weak sequencing is also useful in action refinement, for one might want to refine two consecutive actions into a pair of partially overlapping processes [9,10].

The need for an operator of weak sequential composition becomes most evident when one tries to specify a distributed real-time system (e.g. a telecommunications system) by specifying its legal sequences of events. Therefore such an operator is a welcome constituent of many scenario languages, most notably of message sequence charts [4,7].

When a scenario language is enhanced with new scenario composition operators, it increasingly re-

Email address: monika.kapus-kolar@ijs.si (M. Kapus-Kolar).

sembles a process-algebraic language. The reason is that many process-algebraic languages, e.g. LOTOS [2,1], a standard language for specification of concurrent and reactive systems, have been conceived primarily as languages for abstract specification of process behaviour, i.e. of temporal ordering of events.

For further integration of the scenario and the process-algebraic languages, it is important that the process-algebraic languages adopt all the composition operators defined in the former. In this paper, we suggest how to introduce weak sequencing into E-LOTOS [3,11], the enhanced standard successor of LOTOS. We formalize the proposed approach for discrete-time basic E-LOTOS processes and give informal guidelines for its generalization to full E-LOTOS.

The paper is organized as follows. Sect. 2 is a brief overview of the various kinds of basic E-LOTOS processes. In Sect. 3, we propose how to enhance them with weak sequencing. In Sect. 4, we provide some guidelines for extending the enhancements to processes with data and to a dense-time setting. Sect. 5 concludes the paper.

2. Discrete-time basic E-LOTOS processes

In this section, we briefly describe the discrete-time operational semantics of basic E-LOTOS processes. We use a simplified abstract syntax of processes, with parentheses for unambiguous parsing.

The simplest E-LOTOS process is time block "**block**", a process without any steps.

Process "**stop**" is also inactive, but still willing to age, i.e. execute steps τ reflecting progress of time (Fig. 1).

$\mathbf{stop} \xrightarrow{\tau} \mathbf{stop}$ (IN1)

Fig. 1. Inaction.

Process "**null**" has a single step, a special event δ denoting successful termination (Fig. 2).

$\mathbf{null} \xrightarrow{\delta} \mathbf{block}$ (TM1)

Fig. 2. Successful termination

Let t denote a non-negative, and d a positive integer. A "$\mathbf{wait}(t)$" denotes a process which successfully terminates after t time units (Fig. 3).

$\mathbf{wait}(0) \xrightarrow{\delta} \mathbf{block}$ (WT1)	$\mathbf{wait}(d) \xrightarrow{\tau} \mathbf{wait}(d-1)$ (WT2)

Fig. 3. Waiting.

"**i**" denotes a process which immediately executes an anonymous internal action **i** and then successfully terminates (Fig. 4).

$\mathbf{i} \xrightarrow{\mathbf{i}} \mathbf{null}$ (IA1)

Fig. 4. Internal action.

A "**signal** X" denotes a process which immediately issues a signal X, from a universe $\mathcal{X}$, and then successfully terminates (Fig. 5).

$\mathbf{signal}\ X \xrightarrow{X} \mathbf{null}$ (SG1)

Fig. 5. Signal.

A "G" denotes a process which is at any time ready to execute an interaction G, from a universe $\mathcal{G}$, with its environment on gate G and then successfully terminate (Fig. 6).

$G \xrightarrow{G} \mathbf{null}$ (UG1)	$G \xrightarrow{\tau} G$ (UG2)

Fig. 6. Untimed gate action.

A "$G@t$" is ready for a G followed by δ only t time units after its start (Fig. 7).

$G@0 \xrightarrow{G} \mathbf{null}$ (TG1)	$G@d \xrightarrow{\tau} G@(d-1)$ (TG3)
$G@0 \xrightarrow{\tau} \mathbf{stop}$ (TG2)	

Fig. 7. Timed gate action.

Hence a process step is a τ or an event E. An E is an action A, from a universe $\mathcal{A}$, or a trappable event T. An A is an **i** or a G. A T is a δ or an X. An **i** or a T is by definition urgent, i.e. cannot have τ as an alternative.

A "**trap** T_1 **is** $B_1 \ldots T_n$ **is** B_n **in** B_{n+1}" denotes a process which basically executes B_{n+1}, but if a T_i becomes feasible, this is interpreted as a termination of B_{n+1}, and control is consequently transferred to B_i (Fig. 8). "$B_1; B_2$" is a shorthand for "**trap** δ **is** B_2 **in** B_1", i.e. for sequential composition of a B_1 and a B_2.

$$\frac{B_{n+1} \xrightarrow{E} B'_{n+1}}{\textbf{trap } T_1 \textbf{ is } B_1 \ldots T_n \textbf{ is } B_n \textbf{ in } B_{n+1} \xrightarrow{E} \textbf{trap } T_1 \textbf{ is } B_1 \ldots T_n \textbf{ is } B_n \textbf{ in } B'_{n+1}} \; [E \notin \{T_1, \ldots, T_n\}] \; \text{(TP1)}$$

$$\frac{B_{n+1} \xrightarrow{T_i} \; , \; B_i \xrightarrow{E} B'_i}{\textbf{trap} \ldots T_i \textbf{ is } B_i \ldots \textbf{in } B_{n+1} \xrightarrow{E} B'_i} \; \text{(TP2)} \qquad \frac{B_{n+1} \xrightarrow{T_i} \; , \; B_i \xrightarrow{\tau} B'_i}{\textbf{trap} \ldots T_i \textbf{ is } B_i \ldots \textbf{in } B_{n+1} \xrightarrow{\tau} B'_i} \; \text{(TP4)}$$

$$\frac{B_{n+1} \xrightarrow{\tau} B'_{n+1}}{\textbf{trap } T_1 \textbf{ is } B_1 \ldots T_n \textbf{ is } B_n \textbf{ in } B_{n+1} \xrightarrow{\tau} \textbf{trap } T_1 \textbf{ is } B_1 \ldots T_n \textbf{ is } B_n \textbf{ in } B'_{n+1}} \; \text{(TP3)}$$

Fig. 8. Trapping.

$$B_1[X > B_2 \stackrel{def}{=} B_1[X > (B_2, B_2) \qquad \frac{B_1 \xrightarrow{A} B'_1}{B_1[X > (B_2, B_3) \xrightarrow{A} B'_1[X > (B_2, B_3)} \; \text{(SR1)} \qquad \frac{B_1 \xrightarrow{\delta}}{B_1[X > (B_2, B_3) \xrightarrow{\mathbf{i}} \textbf{null}} \; \text{(SR2)}$$

$$\frac{B_1 \xrightarrow{X'} B'_1}{B_1[X > (B_2, B_3) \xrightarrow{\mathbf{i}} \textbf{signal } X'; (B'_1[X > (B_2, B_3))} \; \text{(SR3)} \qquad \frac{B_2 \xrightarrow{A} B'_2}{B_1[X > (B_2, B_3) \xrightarrow{A} \textbf{trap } X \textbf{ is } B_1[X > B_3 \textbf{ in } B'_2} \; \text{(SR4)}$$

$$\frac{B_2 \xrightarrow{X'} B'_2}{B_1[X > (B_2, B_3) \xrightarrow{\mathbf{i}} \textbf{signal } X'; \textbf{trap } X \textbf{ is } B_1[X > B_3 \textbf{ in } B'_2} \; [X' \neq X] \; \text{(SR5)} \qquad \frac{B_1 \xrightarrow{\tau} B'_1 \; , \; B_2 \xrightarrow{\tau} B'_2}{B_1[X > (B_2, B_3) \xrightarrow{\tau} B'_1[X > (B'_2, B_3)} \; \text{(SR6)}$$

Fig. 9. Suspend/resume.

If an X is trapped, it is because it represents an exception, i.e. an unsuccessful termination of a process. For such an X, "**signal** X" is a misleading specification, and it should better be specified by "**raise** X" (Fig. 10).

$$\textbf{raise } X \xrightarrow{X} \textbf{block} \; \text{(EX1)}$$

Fig. 10. Exception.

If a T has an alternative, its trapping might cause non-deterministic process aging. In E-LOTOS, such aging is strictly forbidden. Therefore the E-LOTOS operational semantics secures that no T ever has an alternative. Let us note that in the adopted interleaving semantics, two events which are concurrent in a particular process state are also alternative next steps of the process.

For a T, to have alternatives means to be in a decisive position. E-LOTOS has many process composition operators potentially able to put a T in such a position. Whenever this happens, a semantic rule prefixes T with an additional **i** taking over the decisive role (e.g. rule CH2 in Fig. 11).

A "$B_1[]B_2$", where it is assumed that neither B_1 nor B_2 can execute δ as its first event, denotes a process behaving as B_1 or as B_2, where the choice is made upon the first event (Fig. 11).

$$\frac{B_i \xrightarrow{A} B'_i}{B_1[]B_2 \xrightarrow{A} B'_i} \; [i \in \{1,2\}] \; \text{(CH1)}$$

$$\frac{B_i \xrightarrow{X} B'_i}{B_1[]B_2 \xrightarrow{\mathbf{i}} \textbf{signal } X; B'_i} \; [i \in \{1,2\}] \; \text{(CH2)}$$

$$\frac{\forall i \in \{1,2\}. B_i \xrightarrow{\tau} B'_i}{B_1[]B_2 \xrightarrow{\tau} B'_1[]B'_2} \; \text{(CH3)}$$

Fig. 11. Choice.

A "$B_1[X > B_2$", where it is assumed that neither δ nor X can be executed by B_2 as its first event, denotes process B_1 potentially suspended upon the first event of B_2. If signal X occurs in B_2 after suspension of B_1, it is implicitly trapped as an exception. Consequently, B_1 is resumed, while B_2 is reset to its initial state, becoming ready for another suspension of B_1. If B_1, while running, becomes ready for a δ, the composite process may execute an **i** leading to successful termination.

This is the semantics of "$B_1[X > B_2$" as described in all tutorial texts on E-LOTOS and formalized in Fig. 9. In the formalization, "$B_1[X > B_2$" is rewritten into "$B_1[X > (B_2, B_2)$", where the first B_2 represents the currently pending instance of B_2, while the second B_2 acts as a constant providing information on what the following instances

$$\frac{\forall i \in \Sigma. B_i \xrightarrow{A} B_i'}{\textbf{par } D \textbf{ in } [\Gamma_1]B_1 \;||\ldots||\; [\Gamma_n]B_n \xrightarrow{A} \textbf{par } D \textbf{ in } [\Gamma_1]B_1' \;||\ldots||\; [\Gamma_n]B_n'} \left[\begin{array}{l}\emptyset \subset \Sigma \subseteq \{1,\ldots,n\}\\ Exec(A,\Sigma,D,\Gamma_1,\ldots,\Gamma_n)\\ \forall i \notin \Sigma.(B_i' = B_i)\end{array}\right] \quad \text{(PR1)}$$

$$\frac{B_i \xrightarrow{X} B_i'}{\textbf{par } D \textbf{ in } [\Gamma_1]B_1 \;||\ldots||\; [\Gamma_n]B_n \xrightarrow{\mathbf{i}} \textbf{signal } X; (\textbf{par } D \textbf{ in } [\Gamma_1]B_1' \;||\ldots||\; [\Gamma_n]B_n')} \left[\begin{array}{l} i \in \{1,\ldots,n\}\\ \forall k \neq i.(B_k' = B_k)\end{array}\right] \quad \text{(PR2)}$$

$$\frac{\forall i \in \{1,\ldots,n\}. B_i \xrightarrow{\delta} B_i'}{\textbf{par } D \textbf{ in } [\Gamma_1]B_1 \;||\ldots||\; [\Gamma_n]B_n \xrightarrow{\delta} \textbf{par } D \textbf{ in } [\Gamma_1]B_1' \;||\ldots||\; [\Gamma_n]B_n'} \quad \text{(PR3)}$$

$$\frac{\forall i \in \Sigma. B_i \xrightarrow{\tau} B_i'\ ,\ \forall i \in (\{1,\ldots,n\}\backslash\Sigma). B_i \xrightarrow{\delta}}{\textbf{par } D \textbf{ in } [\Gamma_1]B_1 \;||\ldots||\; [\Gamma_n]B_n \xrightarrow{\tau} \textbf{par } D \textbf{ in } [\Gamma_1]B_1' \;||\ldots||\; [\Gamma_n]B_n'} \left[\begin{array}{l}\emptyset \subset \Sigma \subseteq \{1,\ldots,n\}\\ \forall i \notin \Sigma.(B_i' = \textbf{null})\end{array}\right] \quad \text{(PR4)}$$

Fig. 12. Parallel composition.

should be. The official formalization of "$B_1[X > B_2$" [3,11] *contains an error*, as if in rule SR6, τ reduced "$B_1[X > (B_2, B_3)$" to "$B_1[X > (B_2', B_2')$".

Let Γ denote a subset of $\mathcal{G}$. A "**par** D **in** $[\Gamma_1]B_1 \;||\ldots||\; [\Gamma_n]B_n$", where each element in the list D is of the form "$G\#N$" with N a positive integer, denotes parallel composition of processes B_1 to B_n (Fig. 12). Each B_i is associated with a Γ_i listing the gates on which B_i synchronizes with its peers (events G not in Γ_i are, like events **i** and X, executed by B_i on its own). If the gate G on which a synchronization occurs has its synchronization degree N defined in D, this is a synchronization of exactly N processes B_i with G in Γ_i, otherwise it is a synchronization of all such processes. Let the policy be encoded as a predicate $Exec(A, \Sigma, D, \Gamma_1, \ldots, \Gamma_n)$ which for every $\Sigma \subseteq \{1, \ldots, n\}$ defines whether it is legal that the composite process executes an A as a common action of exactly the processes B_i with $i \in \Sigma$. The composite process successfully terminates when all its constituents do. "$B_1|[\Gamma]|B_2$" is a shorthand for "**par** $\emptyset$ **in** $[\Gamma]B_1 \;||\; [\Gamma]B_2$", and "$B_1|||B_2$" for pure interleaving "$B_1|[\emptyset]|B_2$".

A "**rename** R **in** B_1", where each element in the list R is of the form "G **is** G'" or "X **is** X'", and R defines at most one new name E' per event E, denotes process B_1 with its events renamed as specified by R (Fig. 13).

$$\begin{array}{l} if\ (E \textbf{ is } E' \in R)\ then\ (R(E) \stackrel{def}{=} E')\\ if\ \nexists E'.(E \textbf{ is } E' \in R)\ then\ (R(E) \stackrel{def}{=} E)\end{array}$$

$$\frac{B_1 \xrightarrow{E} B_1'}{\textbf{rename } R \textbf{ in } B_1 \xrightarrow{R(E)} \textbf{rename } R \textbf{ in } B_1'} \quad \text{(RN1)}$$

$$\frac{B_1 \xrightarrow{\tau} B_1'}{\textbf{rename } R \textbf{ in } B_1 \xrightarrow{\tau} \textbf{rename } R \textbf{ in } B_1'} \quad \text{(RN2)}$$

Fig. 13. Renaming.

A "**hide** Γ **in** B_1" denotes process B_1 with all its G listed in Γ hidden, i.e. converted into an **i** (Fig. 14). Rule HD2 enforces urgency of the new internal events.

$$if\ (E \in \Gamma)\ then\ (E\backslash\Gamma \stackrel{def}{=} \mathbf{i})\ else\ (E\backslash\Gamma \stackrel{def}{=} E)$$

$$\frac{B_1 \xrightarrow{E} B_1'}{\textbf{hide } \Gamma \textbf{ in } B_1 \xrightarrow{E\backslash\Gamma} \textbf{hide } \Gamma \textbf{ in } B_1'} \quad \text{(HD1)}$$

$$\frac{B_1 \xrightarrow{\tau} B_1'\ ,\ \forall G \in \Gamma. B_1 \stackrel{G}{\not\rightarrow}}{\textbf{hide } \Gamma \textbf{ in } B_1 \xrightarrow{\tau} \textbf{hide } \Gamma \textbf{ in } B_1'} \quad \text{(HD2)}$$

Fig. 14. Hiding.

A "P" denotes an instantiation of a process P whose behaviour B_1 is defined by a declaration "P **is** B_1" (Fig. 15). Unguarded recursion leads to time block [6]. Instantiation of formal gates and formal signals need not be considered as a separate issue, because it is nothing but renaming.

$$\frac{B_1 \xrightarrow{E} B_1'}{P \xrightarrow{E} B_1'}\ [P \textbf{ is } B_1] \quad \text{(PI1)} \qquad \frac{B_1 \xrightarrow{\tau} B_1'}{P \xrightarrow{\tau} B_1'}\ [P \textbf{ is } B] \quad \text{(PI2)}$$

Fig. 15. Process instantiation.

A "**loop** X **in** B_1" denotes a process executing a sequence of processes B_1 until signal X occurs in the current B_1 and is interpreted as successful termination of the loop. As "**loop** X **in** B_1" is equivalent to "**trap** X **is null in** P" with P defined as "$B_1; P$", we need not consider it separately.

3. Discrete-time basic E-LOTOS with weak sequencing

3.1. *The necessary additional concepts*

3.1.1. *Untrappable signals*

Process "**signal** X" basically denotes signal X followed by successful termination. However, in a context where X is trapped, this is not the actual role of the process, which must in that case be interpreted as exceptional termination X.

To be able to introduce weak sequencing in the style of [10], we resolve the ambiguity by classifying urgent, non-anonymous, non-δ events into exceptions X, from $\mathcal{X}$, and signals S, from a universe $\mathcal{S}$. Their legal specifications will be "**raise** X" and "**signal** S", while processes "**signal** X" and "**raise** S" are from now on forbidden, and so is trapping of signals. Exceptions, like δ, retain exclusively the role of terminations, implying that we may with no harm pretend that an X always reduces its executor to an equivalent of "**block**".

Hence from now on, $\mathcal{A}$ (the universe of actions) is enhanced with non-anonymous, unsynchronizable, urgent actions S. As signals are, unlike exceptions, untrappable, they can with no harm be in a decisive position, as any other action.

Although signals are unsynchronizable, they are, like gate actions, considered observable, because the environment is allowed to detect them through passive observation [5]. Hence the universe $\mathcal{O}$ of observable actions O is $(\mathcal{G} \cup \mathcal{S})$. We propose to allow hiding for all O, including all S.

3.1.2. *Legal accelerations*

Suppose that for a pair of consecutive processes B and B', it has been specified that it is acceptable for events E' in B' to overtake events E in B. Like [10], we define that E and E' must both belong to $\mathcal{A}$.

We also forbid E and E' to be an **i**, because a process might have internal actions which are not explicitly specified. So if commutation of a pair of actions is to be specified, they must be non-anonymous on the particular specification level, i.e. their hiding (if any) postponed to a higher level.

Hence E and E' can only be an O and an O'. Unlike [10], we do not insist that they must be different. However, if they are, their commutation means that they may as well be executed concurrently.

3.1.3. *Early aging*

Without weak sequencing, a τ step of a B results in aging of all its initial events. In the presence of weak sequencing, it might happen that some initial events of a B are logically enabled, and thus start aging, earlier than the others, implying that τ needs to be furnished with information on the kinds of events of B that it ages. Hence we superscribe τ with a set concisely listing the events.

It turns out that the set can be simply a subset of $\mathcal{A}$, which we shall denote by Ξ. This is because whenever aging is allowed for one kind of non-accelerable events, it is also allowed for all the other kinds, including **i**. Hence if the set is to indicate aging for events of some kind T, it suffices that it contains **i**. In the enhanced semantics, we shall even employ $\tau^{\emptyset}$ steps.

Before we proceed, let us emphasize that for an independent B, one is, as with the original semantics, interested only in its E and $\tau^{\mathcal{A}}$ (i.e. the ordinary τ) steps. However, such a step of a B often consists of various steps of its subprocesses, where a constituent step is not necessarily an E or a $\tau^{\mathcal{A}}$.

3.1.4. *Restricted use of waiting*

With the possibility of early aging, the meaning of "**wait**(t)" is no longer obvious. In a "B; **wait**(t); B'", is it a non-overtakable process between the termination of B and the start of B', or an additional delay for individual events in B' relatively to individual events in B? In the latter case, does it apply to all events of B' or only to its initial events? What if the termination of B is also delayed? To avoid the ambiguities, we define that a "**wait**(t)" may be used exclusively for delaying an individual action, i.e. as a prefix of a "G", a "$G@t'$", an "**i**" or a "**signal** S".

With the above restriction, one can no longer directly use waiting for delaying a T. Fortunately, a

T is never in a decisive position, implying that it can be without a problem prefixed with an auxiliary delayed **i** implementing the desired delay for T.

3.1.5. *Process intentions*

Let F, from a universe $\mathcal{F}$, denote a particular form of process termination. An F can be an ε, denoting non-termination (e.g. because of an infinite run, a deadlock or a premature time block), or a T. Let Φ denote a subset of $\mathcal{F}$.

For a B, let $\mathcal{I}(B)$ denote its current intentions. If $\mathcal{I}(B)$ is an "$\{(F,\Xi_F)|(F \in \Phi)\}$", this means that B intends to conclude by one of the terminations listed in Φ, where for each F in Φ, Ξ_F lists the actions potentially preceding F in B.

In the semantics we propose, computation of the possible future events of a composite process is strictly compositional, to keep its complexity within reasonable limits. Consequently, the computation does not always give precise results. In particular, a Φ without ε does not imply that the process will actually reach a T in Φ. It might as well deadlock, because of insufficient co-operation or a premature time block of its subprocesses or of its environment. However, it is secured that the process will not reach a T outside Φ, or precede an F in Φ by an A not in Ξ_F, or enter a trivially preventable deadlock. The described policy is the same as the one embedded in [10].

The current intentions of a B are important for early aging of actions in the handlers of its terminations. Let O be an action in a B' specified as the handler of a termination T in a B. Early aging of O must be considered as soon as B enters a state in which T is in Φ and every member of Ξ_T is by the specification an action which actions O in B' are allowed to overtake.

3.1.6. *Commitments*

By executing an auxiliary step "$\{(F,\Xi_F)|(F \in \Phi)\}$", a B restricts its future behaviour as much as necessary to become a process with intentions "$\{(F,\Xi_F)|(F \in \Phi)\}$". For an elementary process, the only legal way of deleting a possible run is to refuse execution of a non-urgent action. A commitment made by a composite process will in all cases be implemented simply as suitable commitments of its constituents [10], where we shall try to keep the constituent commitments as mild as possible. When minimization or maximization is required for a particular set, that will be indicated by "*min*" or "*max*", respectively.

Commitments are necessary in accelerated action execution. Whenever an O directly or indirectly guarded by a B is executed before B terminates in a way justifying the action, B must simultaneously make a commitment restricting its future behaviour to the runs legalizing the O [10]. If B then executes such a run, but unexpectedly deadlocks, this is, like in [10], not considered a flaw. Successive accelerated action executions might require B to make more and more restrictive commitments.

3.2. *Enhanced semantics of individual process types*

3.2.1. *Time block*

A B specified as "**block**" behaves as defined in Fig. 16.

$\mathcal{I}(\mathbf{block}) \stackrel{def}{=} \{(\varepsilon,\emptyset)\}$	$\mathbf{block} \xrightarrow{\tau^{\Xi}} \mathbf{block}$ $[\Xi \neq \mathcal{A}]$	(TB1)
	$\mathbf{block} \xrightarrow{\{(\varepsilon,\emptyset)\}} \mathbf{block}$	(TB2)

Fig. 16. Additional rules for time block.

B can participate in aging, but not in the ordinary (non-selective) aging $\tau^{\mathcal{A}}$ (TB1).

B can commit to its current behaviour (TB2).

3.2.2. *Inaction*

A B specified as "**stop**" behaves as defined in Fig. 17.

$\mathcal{I}(\mathbf{stop}) \stackrel{def}{=} \{(\varepsilon,\emptyset)\}$	$\mathbf{stop} \xrightarrow{\tau^{\Xi}} \mathbf{stop}$	(IN1')
	$\mathbf{stop} \xrightarrow{\{(\varepsilon,\emptyset)\}} \mathbf{stop}$	(IN2)

Fig. 17. Modified and additional rules for inaction.

Rule IN1' is an analogue of rule IN1 in Fig. 1 and defines that B can participate in aging of any kind, particularly in steps $\tau^{\mathcal{A}}$.

B can commit to its current behaviour (IN2).

3.2.3. *Successful termination*

A B specified as "**null**" behaves as defined in Figs. 2 and 18.

$\mathcal{I}(\textbf{null}) \stackrel{def}{=} \{(\delta, \emptyset)\}$	$\textbf{null} \stackrel{\tau^{\Xi}}{\longrightarrow} \textbf{null}$ $[\Xi \neq \mathcal{A}]$ (TM2)
	$\textbf{null} \stackrel{\{(\delta,\emptyset)\}}{\longrightarrow} \textbf{null}$ (TM3)

Fig. 18. Additional rules for successful termination

B can execute a δ and then block (TM1).

B can participate in aging, but not in the ordinary aging $\tau^{\mathcal{A}}$ (TM2).

B can commit to its current behaviour (TM3).

3.2.4. *Internal actions*

A B specified as "**i**" behaves as defined in Figs. 4 and 19.

$\mathcal{I}(\textbf{i}) \stackrel{def}{=} \{(\delta, \{\textbf{i}\})\}$	$\textbf{i} \stackrel{\tau^{\Xi}}{\longrightarrow} \textbf{i}$ $[\textbf{i} \notin \Xi]$ (IA2)	$\textbf{i} \stackrel{\{(\delta,\{\textbf{i}\})\}}{\longrightarrow} \textbf{i}$ (IA3)

Fig. 19. Additional rules for an internal action.

B can execute an **i** and become a "**null**" (IA1).

B can participate in aging, but only as long as it does not include aging of **i**, as $\tau^{\mathcal{A}}$ does (IA2).

B can commit to its current behaviour (IA3).

3.2.5. *Signals*

A B specified as a "**signal** S" behaves as defined in Fig. 20.

$\mathcal{I}(\textbf{signal } S) \stackrel{def}{=} \{(\delta, \{S\})\}$
$\textbf{signal } S \stackrel{S}{\longrightarrow} \textbf{null}$ (SG1')
$\textbf{signal } S \stackrel{\tau^{\Xi}}{\longrightarrow} \textbf{signal } S$ $[S \notin \Xi]$ (SG2)
$\textbf{signal } S \stackrel{\{(\delta,\{S\})\}}{\longrightarrow} \textbf{signal } S$ (SG3)

Fig. 20. Modified and additional rules for a signal.

Rule SG1' is an analogue of rule SG1 in Fig. 5 and defines that B can execute an S and become a "**null**".

B can participate in aging, but only as long as it does not include aging of S, as $\tau^{\mathcal{A}}$ does (SG2).

B can commit to its current behaviour (SG3).

3.2.6. *Exceptions*

A B specified as a "**raise** X" behaves as defined in Figs. 10 and 21.

B can execute an X and then block (EX1).

$\mathcal{I}(\textbf{raise } X) \stackrel{def}{=} \{(X, \emptyset)\}$
$\textbf{raise } X \stackrel{\tau^{\Xi}}{\longrightarrow} \textbf{raise } X$ $[\Xi \neq \mathcal{A}]$ (EX2)
$\textbf{raise } X \stackrel{\{(X,\emptyset)\}}{\longrightarrow} \textbf{raise } X$ (EX3)

Fig. 21. Additional rules for an exception.

B can participate in aging, but not in the ordinary aging $\tau^{\mathcal{A}}$ (EX2).

B can commit to its current behaviour (EX3).

3.2.7. *Untimed gate actions*

A B specified as a "G" behaves as defined in Fig. 22 and by rule UG1 in Fig. 6.

$\mathcal{I}(G) \stackrel{def}{=} \{(\delta, \{G\}), (\varepsilon, \emptyset)\}$	$\mathcal{I}(!G) \stackrel{def}{=} \{(\delta, \{G\})\}$
$G \stackrel{\tau^{\Xi}}{\longrightarrow} G$ (UG2')	$!G \stackrel{G}{\rightarrow} \textbf{null}$ (!UG1)
$G \stackrel{\{(\delta,\{G\}),(\varepsilon,\emptyset)\}}{\longrightarrow} G$ (UG3)	$!G \stackrel{\tau^{\Xi}}{\longrightarrow} !G$ (!UG2')
$G \stackrel{\{(\varepsilon,\emptyset)\}}{\longrightarrow} \textbf{stop}$ (UG4)	$!G \stackrel{\{(\delta,\{G\})\}}{\longrightarrow} !G$ (!UG5)
$G \stackrel{\{(\delta,\{G\})\}}{\longrightarrow} !G$ (UG5)	

Fig. 22. Modified and additional rules for an untimed gate action.

B can execute a G and become a "**null**" (UG1).

Rule UG2' is an analogue of rule UG2 in Fig. 6 and defines that B can participate in aging of any kind, particularly in steps $\tau^{\mathcal{A}}$. B can even idle for ever.

B can commit to its current behaviour (UG3).

B can commit to refuse G (UG4).

B can commit to execution of G, thereby reducing to process "$!G$" unable to commit to refusal of G (UG5).

3.2.8. *Timed gate actions*

A B specified as a "$G@t$" behaves as defined in Fig. 24 and by rule TG1 in Fig. 7. In Fig. 24, rules TG2' and TG3', respectively, are analogues of rules TG2 and TG3 in Fig. 7.

If immediate execution of G is specified, B can execute a G and become a "**null**" (TG1).

Rules TG2', TG3' and TG4 define that B can participate in aging of any kind, particularly in steps $\tau^{\mathcal{A}}$. B can even idle for ever. A time step influences B only if it includes aging of actions G. If the timer has already expired, B becomes a

$$\mathcal{I}(B_1[]B_2) \stackrel{def}{=} \{(F,\Xi_F)|(F \in \Phi)\}\ where\ \forall i \in \{1,2\}.(\{(F,\Xi^i_F)|(F \in \Phi_i)\} := \mathcal{I}(B_i))\ ,$$
$$\forall(\{i,j\} = \{1,2\}).(\Phi'_i := \{F|(F \in \Phi_i) \wedge ((F \neq \varepsilon) \vee (\Xi^i_F \neq \emptyset) \vee (\varepsilon \in \Phi_j))\})$$
$$\Phi = \Phi'_1 \cup \Phi'_2\ ,\ \forall F \in \Phi.(\Xi_F = \{A|\exists i \in \{1,2\}.((F \in \Phi'_i) \wedge ((A \in \Xi^i_F) \vee ((A = \mathbf{i}) \wedge (F \neq \varepsilon) \wedge (\Xi^i_F = \emptyset))))\})$$

$$\frac{B_i \stackrel{X}{\rightarrow}}{B_1[]B_2 \stackrel{\mathbf{i}}{\rightarrow} \mathbf{raise}\ X}\ [i \in \{1,2\}] \quad \text{(CH2')}$$

$$\frac{\forall i \in \{1,2\}.B_i \stackrel{\tau^\Xi}{\longrightarrow} B'_i}{B_1[]B_2 \stackrel{\tau^\Xi}{\longrightarrow} B'_1[]B'_2} \quad \text{(CH3')}$$

$$\frac{\forall i \in \{1,2\}.B_i \stackrel{\{(F,\Xi^i_F)|(F\in\Phi_i)\}}{\longrightarrow} B'_i}{B_1[]B_2 \stackrel{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} B'_1[]B'_2} \left[\begin{array}{l}\{(F,\Xi_F)|(F \in \Phi)\} = \mathcal{I}(B'_1[]B'_2) \\ \forall i \in \{1,2\}.(max(\Phi_i \cap \Phi) \wedge \forall F \in \Phi_i.max(\Xi^i_F))\end{array}\right] \quad \text{(CH4)}$$

Fig. 23. Modified and additional rules for choice.

$$\mathcal{I}(G@t) \stackrel{def}{=} \{(\delta,\{G\}),(\varepsilon,\emptyset)\}$$

$$G@0 \stackrel{\tau^\Xi}{\longrightarrow} \mathbf{stop}\ [G \in \Xi] \quad \text{(TG2')}$$
$$G@d \stackrel{\tau^\Xi}{\longrightarrow} G@(d-1)\ [G \in \Xi] \quad \text{(TG3')}$$
$$G@t \stackrel{\tau^\Xi}{\longrightarrow} G@t\ [G \notin \Xi] \quad \text{(TG4)}$$
$$G@t \stackrel{\{(\delta,\{G\}),(\varepsilon,\emptyset)\}}{\longrightarrow} G@t \quad \text{(TG5)}$$
$$G@t \stackrel{\{(\varepsilon,\emptyset)\}}{\longrightarrow} \mathbf{stop} \quad \text{(TG6)}$$
$$G@t \stackrel{\{(\delta,\{G\})\}}{\longrightarrow} !G@t \quad \text{(TG7)}$$

$$\mathcal{I}(!G@t) \stackrel{def}{=} \{(\delta,\{G\})\}$$

$$!G@0 \stackrel{G}{\rightarrow} \mathbf{null} \quad \text{(!TG1)}$$
$$!G@0 \stackrel{\tau^\Xi}{\longrightarrow} \mathbf{stop}\ [G \in \Xi] \quad \text{(!TG2')}$$
$$!G@d \stackrel{\tau^\Xi}{\longrightarrow} !G@(d-1)\ [G \in \Xi] \quad \text{(!TG3')}$$
$$!G@t \stackrel{\tau^\Xi}{\longrightarrow} !G@t\ [G \notin \Xi] \quad \text{(!TG4)}$$
$$!G@t \stackrel{\{(\delta,\{G\})\}}{\longrightarrow} !G@t \quad \text{(!TG7)}$$

Fig. 24. Modified and additional rules for a timed gate action.

"**stop**" (TG2'). If the timer has not yet expired, it is decreased (TG3').

B can commit to its current behaviour (TG5).

B can commit to refuse G (TG6).

B can commit to execution of G, thereby reducing to process "$!G@t$" unable to commit to refusal of G (TG7).

3.2.9. *Waiting*

A B specified as a "$\mathbf{wait}(t); B_1$", where we assume that B_1 is an "**i**", a "**signal** S", a "G" or a "$!G$", behaves as defined in Fig. 25, while a '$\mathbf{wait}(t); G@t'$" and a "$\mathbf{wait}(t); !G@t'$" are equivalent to "$G@(t+t')$" and "$!G@(t+t')$", respectively. In Fig. 25, rules WT1' and WT2', respectively, are analogues of rules WT1 and WT2 in Fig. 3.

$$\mathcal{I}(\mathbf{wait}(t); B_1) \stackrel{def}{=} \mathcal{I}(B_1)$$

$$\frac{B_1 \stackrel{A}{\rightarrow} B'_1}{\mathbf{wait}(0); B_1 \stackrel{A}{\rightarrow} B'_1} \quad \text{(WT1')} \qquad \frac{B_1 \stackrel{\tau^\Xi}{\longrightarrow} B'_1}{\mathbf{wait}(0); B_1 \stackrel{\tau^\Xi}{\longrightarrow} B'_1} \quad \text{(WT4)}$$

$$\frac{B_1 \stackrel{A}{\rightarrow}}{\mathbf{wait}(d); B_1 \stackrel{\tau^\Xi}{\longrightarrow} \mathbf{wait}(d-1); B_1}\ [A \in \Xi] \quad \text{(WT2')}$$

$$\frac{B_1 \stackrel{A}{\rightarrow}}{\mathbf{wait}(d); B_1 \stackrel{\tau^\Xi}{\longrightarrow} \mathbf{wait}(d); B_1}\ [A \notin \Xi] \quad \text{(WT3)}$$

$$\frac{B_1 \stackrel{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} B'_1}{\mathbf{wait}(t); B_1 \stackrel{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} \mathbf{wait}(t); B'_1} \quad \text{(WT5)}$$

Fig. 25. Modified an additional rules for waiting.

If the delay is 0, B can execute the action specified by B_1 (WT1').

If the delay is non-zero, B can participate in aging of any kind. If the aging includes aging of the action specified by B_1, the delay is decreased (WT2'), otherwise it is not (WT3).

If the delay is 0, aging of B is aging of B_1 (WT4).

B can commit to particular terminations and to particular actions preceding them so that B_1 makes the commitment (WT5).

3.2.10. *Choice*

A B specified as a "$B_1[]B_2$" behaves as defined in Fig. 23 and by rule CH1 in Fig. 11. In Fig. 23, rules CH2' and CH3', respectively, are analogues of rules CH2 and CH3 in Fig. 11.

An A resolves the choice in favour of its executor B_i (CH1).

$$\mathcal{I}(\textbf{trap } T_1|C_1 \textbf{ is } B_1 \ldots T_n|C_n \textbf{ is } B_n \textbf{ in } B_{n+1}) \overset{def}{=} \{(F,\Xi_F)|(F \in \Phi)\} \textit{ where}$$
$$\forall i \in \{1,\ldots,(n+1)\}.(\{(F,\Xi^i_F)|(F \in \Phi_i)\} := \mathcal{I}(B_i)) \ , \ \Phi = (\cup_{(i \in \{1,\ldots,n\}) \wedge (T_i \in \Phi_{n+1})} \Phi_i) \cup (\Phi_{n+1} \backslash \{T_1,\ldots,T_n\}) \ ,$$
$$\forall F \in \Phi.(\Xi_F = \{A|(\exists i \in \{1,\ldots,n\}.((T_i \in \Phi_{n+1}) \wedge (F \in \Phi_i) \wedge (A \in (\Xi^{n+1}_{T_i} \cup \Xi^i_F)))) \vee ((F \in (\Phi_{n+1} \backslash \{T_1,\ldots,T_n\})) \wedge (A \in \Xi^{n+1}_F))\})$$

$$\frac{B_{n+1} \xrightarrow{E} B'_{n+1}}{\textbf{trap } T_1|C_1 \textbf{ is } B_1 \ldots T_n|C_n \textbf{ is } B_n \textbf{ in } B_{n+1} \xrightarrow{E} \textbf{trap } T_1|C_1 \textbf{ is } B_1 \ldots T_n|C_n \textbf{ is } B_n \textbf{ in } B'_{n+1}} [E \notin \{T_1,\ldots,T_n\}] \quad \text{(TP1')}$$

$$\frac{B_{n+1} \xrightarrow{T_i} \ , \ B_i \xrightarrow{E} B'_i}{\textbf{trap} \ldots T_i|C_i \textbf{ is } B_i \ldots \textbf{in } B_{n+1} \xrightarrow{E} B'_i} \quad \text{(TP2')}$$

$$\frac{B_{n+1} \xrightarrow{T_i} \ , \ B_i \xrightarrow{\tau^{\Xi}} B'_i}{\textbf{trap} \ldots T_i|C_i \textbf{ is } B_i \ldots \textbf{in } B_{n+1} \xrightarrow{\tau^{\Xi}} B'_i} \quad \text{(TP4')}$$

$$\frac{\forall i \in \{1,\ldots,n\}.B_{n+1} \overset{T_i}{\not\longrightarrow} \ , \ \forall i \in \{1,\ldots,(n+1)\}.B_i \xrightarrow{\tau^{\Xi_i}} B'_i}{\textbf{trap } T_1|C_1 \textbf{ is } B_1 \ldots T_n|C_n \textbf{ is } B_n \textbf{ in } B_{n+1} \xrightarrow{\tau^{\Xi_{n+1}}} \textbf{trap } T_1|C_1 \textbf{ is } B'_1 \ldots T_n|C_n \textbf{ is } B'_n \textbf{ in } B'_{n+1}} \left[\begin{array}{l} \{(F,\Xi_F)|(F \in \Phi)\} := \mathcal{I}(B_{n+1}) \\ \forall i \in \{1,\ldots,n\}. \\ \quad (\Xi_i = \{O|(O \in \Xi_{n+1}) \wedge (T_i \in \Phi) \wedge \\ \qquad \forall A \in \Xi_{T_i}.((A,O) \in C_i)\}) \end{array}\right] \quad \text{(TP3')}$$

$$\frac{B_{n+1} \overset{T_i}{\not\rightarrow} \ , \ B_{n+1} \xrightarrow{\{(T_i,\Xi)\}} B'_{n+1} \ , \ B_i \xrightarrow{O} B'_i}{\textbf{trap} \ldots T_i|C_i \textbf{ is } B_i \ldots \textbf{in } B_{n+1} \xrightarrow{O} \textbf{trap } T_i|C_i \textbf{ is } B'_i \textbf{ in } B'_{n+1}} \left[\begin{array}{l} \forall A \in \Xi.((A,O) \in C_i) \\ max(\Xi) \end{array}\right] \quad \text{(TP5)}$$

$$\frac{\forall i \in \{1,\ldots,(n+1)\}.B_i \xrightarrow{\{(F,\Xi^i_F)|(F \in \Phi_i)\}} B'_i}{\textbf{trap } T_1|C_1 \textbf{ is } B_1 \ldots T_n|C_n \textbf{ is } B_n \textbf{ in } B_{n+1} \xrightarrow{\{(F,\Xi_F)|(F \in \Phi)\}} \textbf{trap } T_1|C_1 \textbf{ is } B'_1 \ldots T_n|C_n \textbf{ is } B'_n \textbf{ in } B'_{n+1}} \left[\begin{array}{l} \{(F,\Xi_F)|(F \in \Phi)\} = \\ \quad \mathcal{I}(\textbf{trap } T_1|C_1 \textbf{ is } B'_1 \ldots T_n|C_n \textbf{ is } B'_n \textbf{ in } B'_{n+1}) \\ \forall i \in \{1,\ldots,(n+1)\}.(max(\Phi_i) \wedge \forall F \in \Phi_i.max(\Xi^i_F)) \end{array}\right] \quad \text{(TP6)}$$

Fig. 26. Enhanced semantics of trapping.

When one of the alternatives has an X pending, B can execute an **i** leading to exceptional termination X (CH2').

Aging of B requires that both alternatives age in the particular manner (CH3').

Rule CH4 defines how B commits to particular terminations and to particular events preceding them. Such a commitment might reduce an alternative to such an extent that it can no longer become the selected one.

Example 1 *Commitment "$\{(\delta,G_1)\}$" reduces process "$G_1[]G_2$" to "$!G_1[]$**stop**", which is with respect to E and τ^A steps equivalent to "G_1".*

3.2.11. *Trapping*

Trapping (Fig. 8) is the main sequencing operator of basic E-LOTOS. In Fig. 26, it is generalized into "**trap** $T_1|C_1$ **is** $B_1 \ldots T_n|C_n$ **is** B_n **in** B_{n+1}", where for each T_i, C_i lists the pairs (O,O') such that actions O' in B_i are allowed to overtake actions O in B_{n+1} proceeding towards T_i. The default C_i is empty, as for example in the case of "$B_1;B_2$". In Fig. 26, rules TP1' to TP4', respectively, are analogues of rules TP1 to TP4 in Fig. 8.

If B_{n+1} executes a non-trapped E, this has no side-effects (TP1').

If B_{n+1} has reached a trapped termination T_i, any step of B_i, either an E (TP2') or a τ^{Ξ} (TP4'), transfers control to the handler of T_i.

As long as B_{n+1} does not reach a trapped termination, the composite process B ages so that B_{n+1} ages, i.e. executes a $\tau^{\Xi_{n+1}}$ (TP3'). Besides, for each particular O, it might be that the fact that O is in Ξ_{n+1}, i.e. that the step includes aging of the initial O in B, implies not only that it includes aging of the initial O in B_{n+1}, but also of the initial O in one or more of the termination handlers B_i. Early aging applies to an initial O of a B_i provided that T_i is a potential termination of B_{n+1} and that it is obvious that B_{n+1} will not be precede T_i by an action not overtakable by O.

It might be that B_{n+1} has several potential terminations, where early aging in a termination handler B_i does not imply that T_i will actually be the selected termination of B_{n+1}. However, early aging is important, because when a particular T_i is selected, its handler B_i must be in a state in which it is known that some of its actions were logically

$$\begin{aligned}
&\mathcal{I}(\mathbf{par}\ D\ \mathbf{in}\ [\Gamma_1]B_1\ ||\ldots||\ [\Gamma_n]B_n) \stackrel{def}{=} \{(F,\Xi_F)|(F\in\Phi)\}\ where\ \forall i\in\{1,\ldots,n\}.(\{(F,\Xi^i_F)|(F\in\Phi_i)\} := \mathcal{I}(B_i))\ ,\\
&\quad \Phi = \{F|((F=\delta)\wedge\forall i\in\{1,\ldots,n\}.(F\in\Phi_i))\vee((F\in\mathcal{X})\wedge\exists i\in\{1,\ldots,n\}.(F\in\Phi_i))\vee\\
&\qquad\qquad ((F=\varepsilon)\wedge(\exists i\in\{1,\ldots,n\}.(\varepsilon\in\Phi_i))\wedge\forall i\in\{1,\ldots,n\}.((\varepsilon\in\Phi_i)\vee(\delta\in\Phi_i)))\}\ ,\\
&\quad \forall F\in\Phi.(\Xi_F = \{A|((F\in\mathcal{X})\wedge(A=\mathbf{i}))\vee\\
&\qquad\qquad\qquad \exists k\in\{1,\ldots,n\},\Sigma\subseteq\{1,\ldots,n\}.\\
&\qquad\qquad\qquad ((F\in\Phi_k)\wedge Exec(A,\Sigma,D,\Gamma_1,\ldots,\Gamma_n)\wedge\\
&\qquad\qquad\qquad \forall i\in\Sigma.(((i=k)\wedge(A\in\Xi^i_F))\vee\\
&\qquad\qquad\qquad\qquad ((i\neq k)\wedge\exists F'\in\Phi_i.((A\in\Xi^i_{F'})\wedge((F'=\delta)\vee((F\neq\delta)\wedge((F'=\varepsilon)\vee(F\neq\varepsilon))))))))\})
\end{aligned}$$

$$\frac{B_i \stackrel{X}{\rightarrow}}{\mathbf{par}\ D\ \mathbf{in}\ldots||\ [\Gamma_i]B_i\ ||\ldots \stackrel{\mathbf{i}}{\rightarrow} \mathbf{raise}\ X}\ \text{(PR2')}$$

$$\frac{\forall i\in\Sigma.B_i \stackrel{\tau^{\Xi}}{\longrightarrow} B'_i\ ,\ \forall i\in(\{1,\ldots,n\}\backslash\Sigma).B_i \stackrel{\delta}{\rightarrow}}{\mathbf{par}\ D\ \mathbf{in}\ [\Gamma_1]B_1\ ||\ldots||\ [\Gamma_n]B_n \stackrel{\tau^{\Xi}}{\longrightarrow} \mathbf{par}\ D\ \mathbf{in}\ [\Gamma_1]B'_1\ ||\ldots||\ [\Gamma_n]B'_n} \begin{bmatrix}\emptyset\subset\Sigma\subseteq\{1,\ldots,n\}\\ \forall i\notin\Sigma.(B'_i=\mathbf{null})\end{bmatrix}\text{(PR4')}$$

$$\frac{\forall i\in\{1,\ldots,n\}.B_i \stackrel{\{(F,\Xi^i_F)|(F\in\Phi_i)\}}{\longrightarrow} B'_i}{\mathbf{par}\ D\ \mathbf{in}\ [\Gamma_1]B_1\ ||\ldots||\ [\Gamma_n]B_n \stackrel{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} \mathbf{par}\ D\ \mathbf{in}\ [\Gamma_1]B'_1\ ||\ldots||\ [\Gamma_n]B'_n} \begin{bmatrix}\{(F,\Xi_F)|(F\in\Phi)\} = \mathcal{I}(\mathbf{par}\ D\ \mathbf{in}\ [\Gamma_1]B'_1\ ||\ldots||\ [\Gamma_n]B'_n)\\ (\delta\in\Phi)\Rightarrow\forall k\in\{1,\ldots,n\},G\in(\Xi^k_\delta\cap\Gamma_k).\exists\Sigma\subseteq\{1,\ldots,n\}.\\ ((k\in\Sigma)\wedge Exec(G,\Sigma,D,\Gamma_1,\ldots,\Gamma_n)\wedge\\ \forall i\in\Sigma.(G\in\Xi^i_\delta))\\ \forall i\in\{1,\ldots,n\}.(max(\Phi_i)\wedge\forall F\in\Phi_i.max(\Xi^i_F))\end{bmatrix}\ \text{(PR5)}$$

Fig. 27. Modified and additional rules for parallel composition.

enabled, and consequently started to age, already at particular times before T_i.

Example 2 *In "**trap** $\delta|(G_1,G_2)$ **is** $G_2@1[]G_3@1$ **in** $G_1@1$", G_2 is logically enabled (i.e. ages) from the beginning, while G_3 only after G_1. Hence a $\tau^{\mathcal{A}}$ reduces the process to "**trap** $\delta|(G_1,G_2)$ **is** $G_2@0[]G_3@1$ **in** $G_1@0$". A possible next step is G_2 further reducing the process to "**trap** $\delta|(G_1,G_2)$ **is null in** $!G_1@0$".*

As long as B_{n+1} does not reach a trapped termination, it is possible that a termination handler B_i executes an accelerated O (TP5). This requires that B_{n+1} simultaneously promises that it will proceed towards T_i, executing only actions overtakable by O. In doing that, B_{n+1} must maximize Ξ, the set of its possible future actions, i.e. restrict its future behaviour as little as possible. After the commitment, B_{n+1} can no longer terminate with a T different from T_i, hence trapping of such T is no longer necessary.

Rule TP6 defines how B commits to particular terminations and to particular actions preceding them. Such a commitment might be non-deterministic.

Example 3 *In "**trap** $\delta|(G_1,G_3)$ **is** G_3 **in** B" where B is "**trap** X **is** $G_1[]G_2$ **in** $((G_1;$**raise** $X)[]G_1)$", G_3 may be executed immediately, but only if B commits to successfully terminate without executing G_2. In doing that, B has the option to reduce to "**trap** X **is** $G_1[]G_2$ **in** $(($**stop**$;$**raise** $X)[]!G_1)$", to "**trap** X **is** $!G_1[]$**stop in** $((!G_1;$**raise** $X)[]G_1)$" or to "**trap** X **is** $!G_1[]$**stop in** $((G_1;$**raise** $X)[]!G_1)$". The first of the three processes is with respect to E and $\tau^{\mathcal{A}}$ steps equivalent to "G_1", while the other two are equivalent to "$(G_1;G_1)[]G_1$".*

3.2.12. *Parallel composition*

A B specified as a "**par** D **in** $[\Gamma_1]B_1\ ||\ldots||\ [\Gamma_n]B_n$" behaves as defined in Fig. 27 and by rules PR1 and PR3 in Fig. 12. In Fig. 27, rules PR2' and PR4', respectively, are analogues of rules PR2 and PR4 in Fig. 12.

Rule PR1 defines how a subset of the concurrent processes collectively executes an action.

When one of the concurrent processes has an X pending, B can execute an **i** leading to exceptional termination X (PR2').

Successful termination of B results from successful termination of its constituents (PR3).

Aging of B requires that every its constituent which has not yet reached successful termination ages in the particular manner (PR4'). If B is ready for successful termination, the rule implies that it can age in any manner legal for a "**null**".

Example 4 *In "**trap** $\delta|(G_1,G_2)$ **is** B **in** G_1"*

$$B_1[X|C>B_2 \stackrel{def}{=} B_1[X|C>(B_2,B_2)$$

$$\begin{array}{l}\mathcal{I}(B_1[X|C>(B_2,B_3)) \stackrel{def}{=} \{(F,\Xi_F)|(F\in\Phi)\}\ \mathit{where}\ \forall i\in\{1,2,3\}.(\{(F,\Xi^i_F)|(F\in\Phi_i)\} := \mathcal{I}(B_i))\ ,\\ \quad \Phi'_1 := \{F|(F\in\Phi_1)\wedge((F\neq\varepsilon)\vee(\varepsilon\in\Phi_2)\vee((X\in\Phi_2)\wedge((\varepsilon\in\Phi_3)\vee(X\in\Phi_3))))\}\ ,\\ \quad \forall i\in\{1,2\}.(\Phi'_i := \{F|((i=2)\vee(X\in\Phi_2))\wedge(F\in(\Phi_i\setminus\{X\}))\wedge((F\neq\varepsilon)\vee(\Xi^i_F\neq\emptyset))\})\ ,\ \Phi=\Phi'_1\cup\Phi'_2\cup\Phi'_3\ ,\\ \quad \forall F\in\Phi.(\Xi_F=\{A|(\exists i\in\{1,2,3\}.((F\in\Phi'_i)\wedge((A\in\Xi^i_F)\vee((A=\mathbf{i})\wedge(F\neq\varepsilon)\wedge((\Xi^i_F=\emptyset)\vee(i=1))))))\vee\\ \qquad\qquad (\exists i\in\{1,3\}.((F\in\Phi'_i)\wedge(X\in\Phi_2)\wedge((A\in\Xi^2_X)\vee((X\in\Phi_3)\wedge(A\in\Xi^3_X)))))\vee\\ \qquad\qquad (\exists i\in\{2,3\}.((F\in\Phi'_i)\wedge\exists F'\in\Phi_1.(A\in\Xi^1_{F'})))\})\end{array}$$

$$\frac{B_1 \xrightarrow{A} B'_1}{B_1[X|C>(B_2,B_3) \xrightarrow{A} B'_1[X|C>(B_2,B_3)}\ \text{(SR1')}$$

$$\frac{B_1 \xrightarrow{\delta}}{B_1[X|C>(B_2,B_3) \xrightarrow{\mathbf{i}} \mathbf{null}}\ \text{(SR2')}$$

$$\frac{B_1 \xrightarrow{X'}}{B_1[X|C>(B_2,B_3) \xrightarrow{\mathbf{i}} \mathbf{raise}\ X'}\ \text{(SR3')}$$

$$\frac{B_2 \xrightarrow{A} B'_2}{B_1[X|C>(B_2,B_3) \xrightarrow{A} \mathbf{trap}\ X|C\ \mathbf{is}\ B_1[X|C>B_3\ \mathbf{in}\ B'_2}\ \text{(SR4')}$$

$$\frac{B_2 \xrightarrow{X'}}{B_1[X|C>(B_2,B_3) \xrightarrow{\mathbf{i}} \mathbf{raise}\ X'}\ [X'\neq X]\ \text{(SR5')}$$

$$\frac{B_1 \xrightarrow{\tau^{\Xi}} B'_1\ ,\ B_2 \xrightarrow{\tau^{\Xi}} B'_2}{B_1[X|C>(B_2,B_3) \xrightarrow{\tau^{\Xi}} B'_1[X|C>(B'_2,B_3)}\ \text{(SR6')}$$

$$\frac{\forall i\in\{1,2,3\}.B_i \xrightarrow{\{(F,\Xi^i_F)|(F\in\Phi_i)\}} B'_i}{B_1[X>(B_2,B_3) \xrightarrow{\{(F,\Xi_F)|(F\in\Phi)\}} B'_1[X>(B'_2,B'_3)} \begin{bmatrix}\{(F,\Xi_F)|(F\in\Phi)\}=\mathcal{I}(B'_1[X>(B'_2,B'_3))\\ \forall i\in\{1,2,3\}.(max(\Phi_i)\wedge\forall F\in\Phi_i.max(\Xi^i_F))\end{bmatrix}\ \text{(SR7)}$$

Fig. 28. Enhanced semantics of suspend/resume.

where B *is "*$G_2@2|||\mathbf{block}$*",* G_2 *is logically enabled from the beginning. Hence a* $\tau^{\mathcal{A}}$ *reduces the process to "*$\mathbf{trap}\ \delta|(G_1,G_2)\ \mathbf{is}\ G_2@1|||\mathbf{block}\ \mathbf{in}\ G_1$*", where* B *executes a* $\tau^{\mathcal{O}}$*, acceptable for* **block**.

If the second step is G_1 *reducing the process to "*$\mathbf{trap}\ \delta|(G_1,G_2)\ \mathbf{is}\ G_2@1|||\mathbf{block}\ \mathbf{in}\ \mathbf{null}$*", there cannot be another* $\tau^{\mathcal{A}}$*, because for* B*, this would be a* $\tau^{\mathcal{A}}$*, unacceptable for* **block**.

Rule PR5 defines how B commits to particular terminations and to particular events preceding them. As a Ξ_X always contains $\mathbf{i}$, the only useful commitments B can make are those towards δ or ε. Note that although we strive for the mildest possible commitments of the constituent processes, prevention of an unexpected deadlock requires that at least when the processes all commit to proceed towards successful termination, none of them plans actions which the commitments of its peers render trivially unexecutable. In [10], this precaution is also implemented to some extent, and is useful also for prevention of non-deterministic commitments.

Example 5 *In "*$\mathbf{trap}\ \delta|(G_1,G_3)\ \mathbf{is}\ G_3\ \mathbf{in}\ B$*" where* B *is "*$(G_1[](G_1;G_2))|[G_2]|(G_1[]G_2)$*",* G_3 *may be executed immediately, but only if* B *commits to successfully terminate without executing* G_2*, thereby reducing to "*$(!G_1[](\mathbf{stop};G_2))|[G_2]|(!G_1[]\mathbf{stop})$*", which is with respect to* E *and* $\tau^{\mathcal{A}}$ *steps equivalent to "*$G_1|||G_1$*".*

Without deadlock prevention, B *would have the option to reduce to "*$(!G_1[](\mathbf{stop};G_2))|[G_2]|(!G_1[]G_2)$*", to "*$(!G_1[](\mathbf{stop};G_2))|[G_2]|(G_1[]!G_2)$*", to "*$(!G_1[](G_1;!G_2))\ |[G_2]|(!G_1[]\mathbf{stop})$*" or to "*$(G_1[](!G_1;!G_2))|[G_2]|(!G_1[]\mathbf{stop})$*". The last two of the processes are with respect to* E *and* $\tau^{\mathcal{A}}$ *steps equivalent to the deadlockable "*$(G_1[](G_1;\mathbf{stop}))|||G_1$*". As such, they do not sufficiently justify accelerated execution of* G_3*. In the words of [10],* B*'s permission for* G_3 *requires that its both concurrent constituents issue such a permission, i.e. commit to successfully terminate without executing* G_2*.*

Note that the deadlock pends in B *already before it makes the commitment. Still, if the sequencing was strong,* B *would not have to issue the permission before the danger of deadlock was over. With weak sequencing, this is no longer true.*

3.2.13. *Suspend/resume*

A B specified as a "$B_1[X>B_2$" (Fig. 9) contains implicit trapping of X in B_2. By generalizing the trapping by a commutation relation C, we in

$$\mathcal{I}(\textbf{rename } R \textbf{ in } B_1) \stackrel{def}{=} \{(F,\Xi_F)|(F \in \Phi)\} \textit{ where } R(\varepsilon) \stackrel{def}{=} \varepsilon \ , \ \{(F,\Xi^1_F)|(F \in \Phi_1)\} := \mathcal{I}(B_1) \ , \ \Phi = \{R(F)|(F \in \Phi_1)\} \ , \ \forall F \in \Phi.(\Xi_F = \{A|\exists F' \in \Phi_1, A' \in \Xi^1_{F'}.((F = R(F')) \wedge (A = R(A')))\})$$

$$\frac{B_1 \stackrel{\tau^{\{A|(R(A)\in\Xi)\}}}{\longrightarrow} B'_1}{\textbf{rename } R \textbf{ in } B_1 \stackrel{\tau^{\Xi}}{\longrightarrow} \textbf{rename } R \textbf{ in } B'_1} \quad \text{(RN2')}$$

$$\frac{B_1 \stackrel{\{(F,\Xi^1_F)|(F\in\Phi_1)\}}{\longrightarrow} B'_1}{\textbf{rename } R \textbf{ in } B_1 \stackrel{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} \textbf{rename } R \textbf{ in } B'_1} \left[\begin{array}{l}\{(F,\Xi_F)|(F \in \Phi)\} = \mathcal{I}(\textbf{rename } R \textbf{ in } B'_1) \\ max(\Phi_1) \ , \ \forall F \in \Phi_1.max(\Xi^1_F)\end{array}\right] \quad \text{(RN3)}$$

Fig. 29. Modified and additional rules for renaming.

Fig. 28 generalize B into "$B_1[X|C>B_2$". In the figure, rules SR1' to SR6', respectively, are analogues of rules SR1 to SR6 in Fig. 9. $\mathcal{I}(B_1[X|C>(B_2,B_3))$ is defined under the assumption that "$B_1[X|C>(B_2,B_3)$" represents a "$B'_1[X|C>B'_2$" or a derivative of it.

When B_1 executes an A, this has no effect on B_2 (SR1').

When B_1 has a T pending, B can execute an **i** leading to termination T, that might be successful (SR2') or exceptional (SR3').

When B_2 executes an A, B_1 is suspended (SR4').

When B_2 has an X' different from X pending, B can execute an **i** leading to exceptional termination X' (SR5').

Aging of B requires that B_1 and B_2 both age in the particular manner (SR6').

Rule SR7 defines how B commits to particular terminations and to particular events preceding them. Such a commitment might reduce B_1, B_2 or B_3 to such an extent that it can no longer terminate B, or B_2 or B_3 to such an extent that it can no longer suspend B_1 or no longer facilitates its resumption after a suspension.

The primarily intended application of C is to specify that some O in a suspended B_1 are resumed before B_1 as a whole is resumed. If an O is resumed immediately upon suspension of B_1, it is as if O had not been suspended at all, implying that C can help in specifying that some actions in B_1 are non-suspendable (e.g. because of their particular importance).

Example 6 *In* "$(G_1|||G_2)[X|(G_4,G_1)>(G_3;G_4;\textbf{raise } X)$", *a* G_3 *leads to an equivalent of*
"**trap** $X|(G_4,G_1)$ **is**
$(G_1|||G_2)[X|(G_4,G_1)>(G_3;G_4;\textbf{raise } X)$
in $(G_4;\textbf{raise } X)$", *with a possible accelerated* G_1 *leading to an equivalent of*
"**trap** $X|(G_4,G_1)$ **is**
$G_2[X|(G_4,G_1)>(G_3;G_4;\textbf{raise } X)$
in $(!G_4;\textbf{raise } X)$", *as if* G_1 *has not been suspended. On the other hand, execution of* G_2 *is not resumed until* X *is raised after* G_4.

With the help of C, one can also achieve that after suspension of B_1, some O in the next instance of B_2 occur before the current instance of B_2 reaches termination X, implying that C can help in specification of infinite sequences of partially overlapping instances of B_2. As for a suspended B_1, it is not resumed until all the activated instances of B_2 have reached X.

Example 7 *In* "$G_1[X|(G_3,G_2)>(G_2;G_3;\textbf{raise } X)$", *a* G_2 *leads to an equivalent of*
"**trap** $X|(G_3,G_2)$ **is**
$G_1[X|(G_3,G_2)>(G_2;G_3;\textbf{raise } X)$
in $(G_3;\textbf{raise } X)$", *with a possible accelerated* G_2 *leading to an equivalent of*
"**trap** $X|(G_3,G_2)$ **is**
trap $X|(G_3,G_2)$ **is**
$G_1[X|(G_3,G_2)>(G_2;G_3;\textbf{raise } X)$
in $(G_3;\textbf{raise } X)$
in $(!G_3;\textbf{raise } X)$", *with a possible* G_3 *leading to an equivalent of*
"**trap** $X|(G_3,G_2)$ **is**
$G_1[X|(G_3,G_2)>(G_2;G_3;\textbf{raise } X)$
in $(G_3;\textbf{raise } X)$", *with a possible* G_3 *leading to an equivalent of* "$G_1[X|(G_3,G_2)>(G_2;G_3;\textbf{raise } X)$".

3.2.14. *Renaming*

A B specified as a "**rename** R **in** B_1", where each element in the list R is of the form "G **is** G'", "S **is** S'" or "X **is** X'", behaves as defined in Fig. 29 and by rule RN1 in Fig. 13.

B can execute any event of B_1, though renamed as specified by R (RN1).

Rule RN2', an analogue of rule RN2 in Fig. 13,

$$if\ (E \in \Omega)\ then\ (E\backslash\Omega \overset{def}{=} \mathbf{i})\ else\ (E\backslash\Omega \overset{def}{=} E)$$

$$\mathcal{I}(\mathbf{hide}\ \Omega\ \mathbf{in}\ B_1) \overset{def}{=} \{(F,\{A\backslash\Omega|(A \in \Xi_F)\})|(F \in \Phi)\}\ where\ \{(F,\Xi_F)|(F \in \Phi)\} := \mathcal{I}(B_1)$$

$$\frac{B_1 \overset{E}{\to} B_1'}{\mathbf{hide}\ \Omega\ \mathbf{in}\ B_1 \overset{E\backslash\Omega}{\longrightarrow} \mathbf{hide}\ \Omega\ \mathbf{in}\ B_1'} \quad \text{(HD1')}$$

$$\frac{B_1 \overset{\tau^{\{A|(A\backslash\Omega \in \Xi)\}}}{\longrightarrow} B_1'\ ,\ \forall G \in (\Xi \cap \Omega).B_1 \overset{G}{\not\to}}{\mathbf{hide}\ \Omega\ \mathbf{in}\ B_1 \overset{\tau^{\Xi}}{\longrightarrow} \mathbf{hide}\ \Omega\ \mathbf{in}\ B_1'} \quad \text{(HD2')}$$

$$\frac{B_1 \overset{\{(F,\Xi^1_F)|(F\in\Phi)\}}{\longrightarrow} B_1'}{\mathbf{hide}\ \Omega\ \mathbf{in}\ B_1 \overset{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} \mathbf{hide}\ \Omega\ \mathbf{in}\ B_1'} \begin{bmatrix}\{(F,\Xi_F)|(F \in \Phi)\} = \mathcal{I}(\mathbf{hide}\ \Omega\ \mathbf{in}\ B_1') \\ \forall F \in \Phi.max(\Xi^1_F)\end{bmatrix} \quad \text{(HD3)}$$

Fig. 30. Modified and additional rules for hiding.

defines that aging of B requires that all the involved actions age under their original names.

Rule RN3 defines how B commits to particular terminations and to particular events preceding them.

3.2.15. *Hiding*

Let Ω denote a subset of $\mathcal{O}$. A B specified as a "**hide** Ω **in** B_1" behaves as defined in Fig. 30. In the figure, rules HD1' and HD2', respectively, are analogues of rules HD1 and HD2 in Fig. 14.

B can execute any event of B_1, though hidden if it belongs to Ω (HD1').

Aging of B requires that all the involved actions age under their original names (HD2').

Rule HD3 defines how B commits to particular terminations and to particular events preceding them.

3.2.16. *Process instantiation*

A B specified as a "P" defined by a declaration "P **is** B_1" behaves as defined in Fig. 31 and by rule PI1 in Fig. 15. Let us note that in E-LOTOS, declaration of a P is always accompanied by declaration of $\mathcal{G}(P)$ listing its potential G and by $\mathcal{X}(P)$ listing its potential X. With the enhanced semantics of P, one would also declare $\mathcal{S}(P)$ listing its potential S.

In principle, the events of B are supposed to be the events of B_1 (PI1). However, if P is directly or indirectly instantiated in B_1 and the instantiation is not sufficiently guarded, rule PI1 might for a particular E lead to an infinite proof of executability, implying that B will not be able to execute E. Weak sequencing makes the problem even more acute [10].

$$\mathcal{I}(P) \overset{def}{=} \{(F,\Xi_F)|(F \in \Phi)\}\ where$$
$$P\ \mathbf{is}\ B_1\ ,\ \{(F,\Xi_F)|(F \in \Phi)\} = \mathcal{I}(B_1)\ ,$$
$$max(\Phi \cap \{\varepsilon\})\ ,\ min(\Phi\backslash\{\varepsilon\})\ ,\ \forall F \in \Phi.min(\Xi_F)$$

$$\frac{B_1 \overset{\tau^{\Xi}}{\longrightarrow} B_1'}{P \overset{\tau^{\Xi}}{\longrightarrow} B_1'} \begin{bmatrix}P\ \mathbf{is}\ B_1 \\ (\Xi \cap (\mathcal{G}(P) \cup \mathcal{S}(P) \cup \{\mathbf{i}\})) \neq \emptyset\end{bmatrix} \quad \text{(PI2')}$$

$$P \overset{\tau^{\Xi}}{\longrightarrow} P\ [(\Xi \cap (\mathcal{G}(P) \cup \mathcal{S}(P) \cup \{\mathbf{i}\})) = \emptyset] \quad \text{(PI3)}$$

$$\frac{B_1 \overset{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} B_1'}{P \overset{\{(F,\Xi_F)|(F\in\Phi)\}}{\longrightarrow} B_1'} \begin{bmatrix}P\ \mathbf{is}\ B_1 \\ \{(F,\Xi_F)|(F \in \Phi)\} \neq \mathcal{I}(P) \\ \{(F,\Xi_F)|(F \in \Phi)\} = \mathcal{I}(B_1') \\ \forall F \in \Phi.\exists(F,\Xi'_F) \in \mathcal{I}(P). \\ (\Xi_F \subseteq \Xi'_F)\end{bmatrix} \quad \text{(PI4)}$$

$$P \overset{\mathcal{I}(P)}{\longrightarrow} P \quad \text{(PI5)}$$

Fig. 31. Modified and additional rules for process instantiation.

Rule PI2' is an analogue of rule PI2 in Fig. 15 and defines that P in principle ages exactly as B_1. However, as there is again the problem of infinite proofs, the trivial case of aging known to have no impact on B is covered by a separate rule PI3.

B in principle commits to particular terminations and to particular actions preceding them exactly as B_1 (PI4). However, as there is again the problem of infinite proofs, the trivial case where B commits to its current behaviour is covered by a separate rule PI5.

Computing the intentions $\mathcal{I}(P)$ of individual P in a particular system specification, one collects all the semantic rules applying to processes of the system and computes $\mathcal{I}(P)$ simultaneously satisfying all of them. A particular T is included into Φ of a particular $\mathcal{I}(P)$ only if this proves necessary, and the same applies to inclusion of a particular A in a particular Ξ_F. On the other hand, ε is included in a particular $\mathcal{I}(P)$ whenever possible, particularly in the case of unguarded recursion, which is thereby correctly interpreted as non-termination.

Example 8 *If "P* **is signal** *S", $\mathcal{I}(P)$ is $\{(\delta,\{S\})\}$, because it must be equal to $\mathcal{I}($***signal** *S).*

If "P **is signal** *S; P" or if "P* **is signal** *S|||P", $\mathcal{I}(P)$ is $\{(\varepsilon,\{S\})\}$, because no T needs to be in Φ, while ε can and hence must.*

4. Some guidelines for weak sequencing in full E-LOTOS

It seems that embedding of weak sequencing into full E-LOTOS can follow the same principles as for discrete-time basic E-LOTOS, with the following enhancements:

Every observable action O is a $G(\alpha)$ or an $S(\alpha)$, where α is the associated data. In the following, let Q denote a G or an S, and U an **i** or a Q.

A gate action is specified by a "$G\pi@\pi'[\beta]$" denoting any $G(\alpha)$ such that α matches pattern π, its execution time t matches pattern π', and the pair (π,π') satisfies the additional condition β. In other words, such a process denotes choice between various adequate $O@t$. When the choice needs to be restricted to justify accelerated execution of a subsequent action, the process can make the commitment simply by strengthening β.

In the trapping operator (Sect. 3.2.11), it might be necessary that a commutation relation C_i is very large or even infinite. Such a C_i can be efficiently specified by a list of elements of the form "$(Q(\pi),Q'(\pi'))[\beta]$". Actions $Q'(\alpha')$ in B_i may overtake actions $Q(\alpha)$ in B_{n+1} provided that the list contains a "$(Q(\pi),Q'(\pi'))[\beta]$" such that α matches pattern π, α' matches pattern π', and the pair (α,α') satisfies the additional condition β. Such encoding is appropriate also for the C of the suspend/resume operator.

Every T is a $\delta(\alpha)$ or an $X(\alpha)$. In the following, let Z denote a δ or an X, and Y an ε or a Z.

In the trapping operator, when a Z_i is trapped and handled by B_i, the free variables of B_i are, as specified, instantiated with the data α associated with Z_i. In a general case, B_{n+1} chooses between many different α. The consequently large $\mathcal{I}(B_{n+1})$ can be efficiently represented by an "$\{(Y(\pi_Y)[\beta_Y],\zeta_Y)|(Y\in\Upsilon)\}$". An $Y(\alpha)$ (if Y is ε, α is by definition void) is a possible termination of B_{n+1} if Y is in Υ and α matches pattern π_Y and satisfies the additional condition β_Y. A $U(\alpha')$ (if U is **i**, α' is by definition void) is a possible predecessor of such an $Y(\alpha)$ if in ζ_Y, there is a "$U(\pi)[\beta]$" such that α' matches π and the pair (α,α') satisfies the additional condition β.

For each Z_i potentially trapped in a B_{n+1}, the current intentions of B_{n+1} imply an additional restriction on the free variables of B_i, the handler of Z_i. As the restriction influences the ability of B_i for early aging and for accelerated action execution, it must be updated upon every step of B_{n+1}, while in the standard E-LOTOS, it need not be computed until B_{n+1} actually reaches Z_i.

If the time domain is dense, a time step is a "d^{Ξ}", where d denotes its length and Ξ the actions to which the aging applies. As Ξ might be large or even infinite, it requires an efficient encoding, as a list of elements of the form "$U(\pi)[\beta]$". $d^{\mathcal{A}}$ denotes the ordinary time step of length d, while time steps of other kinds represent selective early aging.

In the trapping operator, if at a time t, B_{n+1} initiates a time step of length d, it is required that $\mathcal{I}(B_{n+1})$ is the same at all times t' with $(t\le t'<(t+d))$. Besides, each termination handler B_i must be able to execute its corresponding early-aging step d^{Ξ_i}. If this is not the case, the time step must be adequately shortened. In a rare anomalous case, no sufficiently short step exists, but this is the usual problem with the interleaving semantics for dense time [6].

5. Concluding remarks

In the paper, we have enhanced the E-LOTOS trapping operator (and the trapping embedded in the E-LOTOS suspend/resume operator) with the possibility of specifying that consecutive processes may partially overlap. Developing the enhanced semantics, we had to introduce four new concepts: non-trappable signals, early aging of actions, intention reporting and process commitments. We also had to restrict the use of waiting. With the proposed semantics, we have successfully extended the ideas of [10] to real-time processes and to processes with more that one possible direction of sequential control transfer.

We conclude by summarizing that the proposed enhancements of E-LOTOS would help in the following very frequent situations:

– When some actions of otherwise consecutive processes belong to different locations of a distributed system, it is appropriate to specify that they are naturally concurrent, to emphasize that the process sequencing in its strong form is not trivially implementable in the particular system.
– When some actions of otherwise consecutive processes are not causally related and do not compete for resources, it is desirable to specify that they may be executed concurrently, to emphasize the possibility of accelerated execution.

References

[1] T. Bolognesi and E. Brinksma, Introduction to the ISO specification language LOTOS, Computer Networks and ISDN Systems 14 (1987) 25–59.

[2] ISO, LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, ISO 8807, ISO – Information Processing Systems – Open Systems Interconnection, 1989.

[3] ISO/IEC, Enhancements to LOTOS (E-LOTOS), ISO/IEC 15437, ISO – Information Technology, 2001.

[4] ITU-T, Message Sequence Charts – MSC-2000, ITU-T Recommendation Z.120, November 1999.

[5] M. Kapus-Kolar, Restoring the concept of observability in E-LOTOS, Computer Standards & Interfaces 22 (2000) 55–60.

[6] L. Léonard and G. Leduc, A formal definition of time in LOTOS, Formal Aspects of Computing 10 (1998) 248–266.

[7] S. Mauw and M. A. Reniers, Operational semantics for MSC'96, Computer Networks 31 (1999) 1785–1799.

[8] A. Rensink and H. Wehrheim, Weak sequential composition in process algebras, in Proc. CONCUR'94, LNCS 836, Springer-Verlag Heidelberg, 1994, pp. 226–241.

[9] A. Rensink and H. Wehrheim, Dependency-based action refinement, in Proc. MFCS'97, LNCS 1295, Springer-Verlag Heidelberg, 1997, pp. 468–477.

[10] A. Rensink and H. Wehrheim, Process algebra with action dependencies, Acta Informatica 38 (2001) 155–234.

[11] A. Verdejo, E-LOTOS: Tutorial and Semantics, MS thesis, Universidad Complutense de Madrid, 1999.

Monika Kapus-Kolar received the BS degree in electrical engineering from the University of Maribor, Slovenia, in 1981, and the MS and PhD degrees in computer science from the University of Ljubljana, Slovenia, in 1984 and 1989, respectively. Since 1981 she has been with the Jožef Stefan Institute, Ljubljana, where she is currently a researcher at the Department of Digital Communications and Networks. Her current research interests include formal specification techniques and methods for development of distributed systems and computer networks.