A failures semantics for ET-LOTOS[1]

Steve Schneider
Royal Holloway, University of London

Jeremy Bryans and Jim Davies
University of Reading

Abstract A denotational semantics is presented for ET-LOTOS [LeL94] in the style of semantics for timed CSP, in terms of timed failures with additional information about internal events. This semantics is consistent with the original operational semantics, and is shown to be the weakest congruence stronger than trace equivalence—the weakest useful congruence.

1 Introduction

The LOTOS specification language was developed to support specification and design of distributed and communicating systems. Recently, a number of proposals have arisen for incorporating time into the language [LeL94, QFA93, BLT94], enabling the modelling of quantitative timing behaviour.

A LOTOS specification consists primarily of a description of desirable behaviour. But we also require a notion of how closely any proposed implementation should resemble that ideal. Many approaches require that an implementation should be at least weakly bisimilar to the specification process. Bisimulation equivalence has the pleasing property that it can often be efficiently checked mechanically, and further it usually has a complete set of characterising algebraic laws. Proposals for timed extensions to LOTOS are generally accompanied by appropriate notions of bisimulation.

On the other hand, the kind of distinctions made by bisimulation equivalence are finer than can generally be made by interacting agents, which do not have access to the internal state of their communication partners. In fact, processes which are not bisimilar may still be indistinguishable in any context. This paper takes the view that if no context can tell a specification process from a proposed implementation, then the implementation should be considered to be satisfactory. Since bisimulation congruence is strictly stronger than this notion of equivalence, it is useful as a sufficient condition for equivalence, but should not be considered as a necessary condition.

[1]Presented at the COST 247 workshop on extending process algebras, Brighton July 1994

We follow the CSP approach of [Sch94], and see how it adapts to the LOTOS operators. For definiteness, we will consider the timed LOTOS proposal of [LeL94], as this appears to be the most amenable to this treatment. We begin by requiring that processes which have different traces should be considered distinct, and then search for the weakest congruence over the language that preserves this distinction. We find that the timed failures used in CSP are not quite enough, and that we also require information about the times at which certain internal events are possible, because some LOTOS operators are sensitive to internal events. The result is a fully abstract denotational semantics for timed LOTOS .

Equivalence and congruence

A congruence on a language is an equivalence relation that is preserved by all of the operators of the language. Hence it is preserved by all contexts built from those operators. Given a notion $\equiv$ of equivalence, the corresponding congruence $\cong$ is the weakest congruence (i.e. the one which makes as few distinctions as possible) which implies $\equiv$-equivalence. In other words

$$\forall P, Q \bullet P \cong Q \Rightarrow P \equiv Q$$

and if we have another congruence $\cong'$ for which

$$\forall P, Q \bullet (P \cong' Q \Rightarrow P \equiv Q)$$

then

$$\forall P, Q \bullet P \cong' Q \Rightarrow P \cong Q$$

For example, untimed trace equivalence on processes identifies two processes if any finite sequence of events (any untimed trace) which can be observed during an exection of one may also be observed of the other. Untimed trace congruence (hereafter called trace congruence) for timed CSP turns out to be timed failures equivalence, where two processes are identified if they have the same set of timed failures (see [Sch94] for details).

In this paper we are interested in the weakest congruence stronger than (untimed) trace equivalence. We regard this as the weakest congruence of any practical use—anything weaker will equate processes which have different traces! For timed CSP, this turns out to be the timed failures semantics, and this is in turn equivalent to may testing equivalence [Hen88] on the operational semantics (where two processes are equivalent if the set of tests each may pass is the same). In general however, may testing equivalence might not be a congruence, and in fact it is not a congruence for timed versions of LOTOS.

Basic Timed LOTOS

We focus on Leduc and Léonard's proposal for Basic ET-LOTOS [LeL94]. They present an extension to the language of Basic LOTOS to include timing operators, and give an operational semantics for the language, in terms of event transitions and delay transitions. Processes are defined in terms of those transitions that may be performed.

An event transition is of the form $\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{Q}$. This indicates that process P may immediately perform event a and then behave as process Q. A delay transition is of the form $\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{Q}$. This indicates that process P may evolve in d units of time to process Q.

LOTOS assumes a universal set of gates G. It also makes use of special events $\mathtt{i}, \delta \notin G$, and we will introduce other such special events $\omega \notin G, \upsilon \in G$ in this paper. The event a ranges over $G \cup \{\mathtt{i}, \delta, \omega\}$. A set of gates Γ is a subset of G. The variables t, u range over a set of time variables; d ranges over the set of time values, which is taken to be $[0, \infty) \cup \{\infty\}$, though it is noted in [LeL94] that any commutative monoid with an absorbent (zero) element ∞ for $+$ will serve as a time domain.

The formal semantics of Basic ET-LOTOS is given in [LeL94]; we reproduce it here to keep the paper self-contained.

The syntax is as follows: A process P is defined by an equational definition with a where clause binding a vector of process variables to a vector of process terms:

$$\mathtt{P} := \mathtt{Q} \quad \mathtt{where} \quad \tilde{\mathtt{Y}} := \tilde{\mathtt{Q}}$$

Every Y occuring free in some Q must appear in the vector of process variables $\tilde{\mathtt{Y}}$. As discussed below, the vector of process terms must be time-guarded in the vector of process variables. This constraint is not imposed in [LeL94], but is necessary for the semantic approach taken in this paper.

The terms Q are defined as follows:

$$\begin{array}{lll} \mathtt{Q} & := & \mathtt{stop} \mid \mathtt{exit}\{\mathtt{d}\} \mid \mathtt{a@t}\{\mathtt{d}\}\,;\mathtt{Q} \mid \Delta^{\mathtt{d}}\,\mathtt{Q} \mid \mathtt{Q} \;\Box\; \mathtt{Q} \\ & & \mid \mathtt{Q} \;|[\Gamma]|\; \mathtt{Q} \mid \mathtt{hide}\;\Gamma\;\mathtt{in}\;\mathtt{Q} \mid \mathtt{Q} \gg \mathtt{Q} \mid \mathtt{Q}\;[>\;\mathtt{Q} \mid \mathtt{Y} \end{array}$$

These denote respectively: deadlock; termination at time d; life reducer (or event timeout); delay; choice; concurrency; abstraction; enabling (or sequential composition); disabling; process variable.

If t does not appear free in Q then the term $\mathtt{a}\{\mathtt{d}\}\,;\mathtt{Q}$ may be used as an abbreviation for $\mathtt{a@t}\{\mathtt{d}\}\,;\mathtt{Q}$. The term $\mathtt{a}\,;\mathtt{Q}$ abbreviates $\mathtt{a}\{\infty\}\,;\mathtt{Q}$ for $\mathtt{a} \neq \mathtt{i}$, and $\mathtt{i}\,;\mathtt{Q}$ is used for $\mathtt{i}\{0\}\,;\mathtt{Q}$.

The rules for the operators are given as follows:

$$\frac{}{\mathtt{stop} \xrightarrow{\mathtt{d}} \mathtt{stop}}$$

$$\frac{}{\mathtt{a@t\{d_1\}\,;P} \xrightarrow{\mathtt{a}} \mathtt{[0/t]P}}$$

$$\frac{}{\mathtt{a@t\{d_1+d\}\,;P} \xrightarrow{\mathtt{d}} \mathtt{a@t\{d_1\}\,;[t+d/t]P}}$$

$$\frac{}{\mathtt{a@t\{d_1+d\}\,;P} \xrightarrow{\mathtt{d}} \mathtt{stop}} \quad [\ \mathtt{a} \neq \mathtt{i}, \mathtt{d} > \mathtt{d_1}\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P'}}{\Delta^{0}\,\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P'}}$$

$$\frac{}{\Delta^{\mathtt{d_1+d}}\,\mathtt{P} \xrightarrow{\mathtt{d}} \Delta^{\mathtt{d_1}}\,\mathtt{P}}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{P'}}{\Delta^{0}\,\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{P'}}$$

$$\frac{}{\mathtt{exit\{d_1\}} \xrightarrow{\delta} \mathtt{stop}}$$

$$\frac{}{\mathtt{exit\{d_1+d\}} \xrightarrow{\mathtt{d}} \mathtt{exit\{d_1\}}}$$

$$\frac{}{\mathtt{exit}\{\mathtt{d_1}\} \xrightarrow{\mathtt{d}} \mathtt{stop}} \quad [\ \mathtt{d} > \mathtt{d_1}\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P'}}{\begin{array}{l}\mathtt{P}\ \square\ \mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{P'} \\ \mathtt{Q}\ \square\ \mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P'}\end{array}}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{P'} \qquad\qquad \mathtt{Q} \xrightarrow{\mathtt{d}} \mathtt{Q'}}{\mathtt{P}\ \square\ \mathtt{Q} \xrightarrow{\mathtt{d}} \mathtt{P'}\ \square\ \mathtt{Q'}}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P'}}{\begin{array}{l}\mathtt{P}\ |[\Gamma]|\ \mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{P'}\ |[\Gamma]|\ \mathtt{Q} \\ \mathtt{Q}\ |[\Gamma]|\ \mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{Q}\ |[\Gamma]|\ \mathtt{P'}\end{array}} \quad [\ \mathtt{a} \notin \Gamma \cup \{\delta\}\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P'} \qquad\qquad \mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{Q'}}{\mathtt{P}\ |[\Gamma]|\ \mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{P'}\ |[\Gamma]|\ \mathtt{Q'}} \quad [\ \mathtt{a} \in \Gamma \cup \{\delta\}\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{P'} \qquad\qquad \mathtt{Q} \xrightarrow{\mathtt{d}} \mathtt{Q'}}{\mathtt{P}\ |[\Gamma]|\ \mathtt{Q} \xrightarrow{\mathtt{d}} \mathtt{P'}\ |[\Gamma]|\ \mathtt{Q'}}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P'}}{\mathtt{hide}\ \Gamma\ \mathtt{in}\ \mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{hide}\ \Gamma\ \mathtt{in}\ \mathtt{P'}} \quad [\ \mathtt{a} \notin \Gamma\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P}'}{\mathtt{hide}\ \Gamma\ \mathtt{in}\ \mathtt{P} \xrightarrow{\mathtt{i}} \mathtt{hide}\ \Gamma\ \mathtt{in}\ \mathtt{P}'} \quad [\ \mathtt{a} \in \Gamma\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{P}' \qquad \forall \mathtt{a} \in \Gamma \bullet \neg(\mathtt{P} \xrightarrow{\mathtt{a}}\ )}{\mathtt{hide}\ \Gamma\ \mathtt{in}\ \mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{hide}\ \Gamma\ \mathtt{in}\ \mathtt{P}'}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P}'}{\mathtt{P} \gg \mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{P}' \gg \mathtt{Q}} \quad [\ \mathtt{a} \neq \delta\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\delta} \mathtt{P}'}{\mathtt{P} \gg \mathtt{Q} \xrightarrow{\mathtt{i}} \mathtt{Q}}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{P}' \qquad \neg(\mathtt{P} \xrightarrow{\delta}\ )}{\mathtt{P} \gg \mathtt{Q} \xrightarrow{\mathtt{d}} \mathtt{P}' \gg \mathtt{Q}}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{P}'}{\mathtt{P}\ [>\ \mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{P}'\ [>\ \mathtt{Q}} \quad [\ \mathtt{a} \neq \delta\ ]$$

$$\frac{\mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{Q}'}{\mathtt{P}\ [>\ \mathtt{Q} \xrightarrow{\mathtt{a}} \mathtt{Q}'}$$

$$\frac{\mathtt{P} \xrightarrow{\delta} \mathtt{P}'}{\mathtt{P\ [>\ Q} \xrightarrow{\delta} \mathtt{P}'}$$

$$\frac{\mathtt{P} \xrightarrow{d} \mathtt{P}' \qquad \mathtt{Q} \xrightarrow{d} \mathtt{Q}'}{\mathtt{P\ [>\ Q} \xrightarrow{d} \mathtt{P'\ [>\ Q'}}$$

$$\frac{\begin{array}{l}[g_1/h_1, \ldots, g_n/h_n]\mathtt{P} \xrightarrow{a} \mathtt{P}' \\ \mathtt{Q}[h_1, \ldots, h_n] \ \mathtt{:=}\ \mathtt{P}\end{array}}{\mathtt{Q}[g_1, \ldots, g_n] \xrightarrow{a} \mathtt{P}'}$$

$$\frac{\begin{array}{l}[g_1/h_1, \ldots, g_n/h_n]\mathtt{P} \xrightarrow{d} \mathtt{P}' \\ \mathtt{Q}[h_1, \ldots, h_n] \ \mathtt{:=}\ \mathtt{P}\end{array}}{\mathtt{Q}[g_1, \ldots, g_n] \xrightarrow{d} \mathtt{P}'}$$

Operationally, there is no problem with instant or unguarded recursions; their transitions may be calculated using the rules in exactly the same way as guarded recursions. A definition such as `S := S` is perfectly legitimate, and defines a time-stop process, one that has no event or delay transitions (in contrast to `stop` which can always allow time to pass). A definition such as `S := a@t{1} ; S` is guarded by the event `a`, and can do as many copies of the event `a` at time *0* as the environment is prepared to allow.

However, the standard denotational approach for timed CSP (which is being followed here) would need a significant alteration in order to deal with such process definitions. Recursive definitions are required to take some time to reach their next invocation. This makes the semantics more straightforward, at the expense of expressibility. Thus any recursive definition must be *time-guarded* in the sense that there is some non-zero delay, imposed by a Δ^d construct, guarding any recursive call. (In a mutual recursion, there must be some universal non-zero time guard which holds for each recursive definition.) As a result, no time-stopping process can be written; the examples in the preceding paragraph are all considered to be unguarded in the timed sense.

Traces

The traces of a process may be extracted from its operational semantics by use of the rules below.

$$\frac{}{\langle\rangle \in \mathit{traces}(P)}$$

$$\frac{P \xrightarrow{a} P' \qquad tr \in \mathit{traces}(P')}{\langle a\rangle^\frown tr \in \mathit{traces}(P)} \quad [\ a \neq i\ ]$$

$$\frac{P \xrightarrow{i} P' \qquad tr \in \mathit{traces}(P')}{tr \in \mathit{traces}(P)}$$

$$\frac{P \xrightarrow{d} P' \qquad tr \in \mathit{traces}(P')}{tr \in \mathit{traces}(P)}$$

The set $\mathit{traces}(P)$ is defined to be the smallest set closed under these inference rules.

Timed failures

The set of events Σ includes every gate in G, and the events δ and ω. The internal event i is not in Σ.

A timed event (t, a) is an element of $[0, \infty) \times \Sigma$.

A timed failure is a particular view of an execution of a process, recording what might have been observed by its environment, or by another process. It will contain the events that were performed, the times at which they were performed, and the times at which events were refused by the process. The approach of CSP is to associate each process with the set of those timed failures that are consistent with some run of the process. Throughout this paper, the term *failure* means timed failure.

A timed trace is a finite sequence of timed events in which the times are non-decreasing. The set of timed traces is denoted TT.

A timed refusal is a set of timed events.

A timed failure is a (timed trace, timed refusal) pair. This is considered a record of an execution; the timed trace records those timed events that were performed, and the timed refusal records those timed events that were refused by the process, during the execution.

Since process executions are finitely variable, in that only finitely many changes of state may occur in a finite interval, it is sufficient to consider timed refusal sets which are finitely variable; where the set of events being refused changes only finitely often. Timed refusals thus have a particular structure: a finite union of refusal tokens.

$$\begin{aligned} \mathit{FRTOK} &= \{[b, e) \times A \mid 0 \le b < e < \infty, A \subseteq^{\mathit{fin}} \Sigma\} \\ \mathit{FRSET} &= \{\bigcup B \mid B \subseteq^{\mathit{fin}} \mathit{FRTOK}\} \end{aligned}$$

Notation

We use s to range over TT, the set of timed traces, and X to range over FRSET, the set of timed refusals; t, u range over the finite times $[0, \infty)$; I ranges over subsets of $[0, \infty)$ (usually intervals).

We use the following operations on sequences of timed events: $s_1 \frown s_2$ denotes the concatenation of s_1 and s_2; $\mathit{end}(s \frown \langle (t, a) \rangle) = t$.

The following projections on sequences are defined by list comprehension:

$$\begin{aligned} s \uparrow I &= \langle (t, a) \mid (t, a) \leftarrow s, t \in I \rangle \\ s \setminus A &= \langle (t, a) \mid (t, a) \leftarrow s, a \notin A \rangle \\ s \restriction A &= \langle (t, a) \mid (t, a) \leftarrow s.a \in A \rangle \\ s + u &= \langle (t + u, a) \mid (t, a) \leftarrow s \rangle \\ \mathrm{events}(s) &= \langle a \mid (t, a) \leftarrow s \rangle \\ \sigma(s) &= \{a \mid s \restriction \{a\} \neq \langle \rangle\} \end{aligned}$$

We also define a number of projections on refusal sets:

$$\begin{aligned} X \lhd u &= \{(t, a) \mid (t, a) \in X, t < u\} \\ X \uparrow I &= \{(t, a) \mid (t, a) \in X, t \in I\} \end{aligned}$$

$$
\begin{aligned}
X - u &= \{(t-u, a) \mid (t,a) \in X, t \geq u\} \\
\sigma(X) &= \{a \mid (u,a) \in X\} \\
end(X) &= sup\{u \mid (u,a) \in X\}
\end{aligned}
$$

We define $end(s, X) = \max\{end(s), end(X)\}$.

Example

The timed failure

$$(\langle(3, a), (8, b)\rangle, [0, 3) \times \{b\} \cup [3, 5) \times \{a\})$$

corresponds to an execution in which the event b was offered to the process but not accepted (i.e. refused) up until time 3; the event a was then observed at time 3; a further copy of the event a was refused over the interval $[3, 5)$; and finally the event b was performed at time 8.

This would be a possible observation of the process a ; Δ^2 b ; stop. The event a can occur at any time, or not at all. If it occurs at time 3, then no further copies of it are possible, and so will be refused over the interval $[3, 5)$. The event b is not enabled in this execution until time 5, so it can be refused over the interval $[0, 3)$. It may be performed at any time after time 5; in this instance, it is performed at time 8.

Extracting failures

We can define inductively what it means for a timed failure to be consistent with a given execution. The relation **exhibits** is a predicate on processes and timed failures drawn from $TT \times FRSET$. The events that may appear in timed traces are $G \cup \{\delta, \omega\}$—any action of interest apart from i.

$$\frac{}{\mathtt{P}\ \textbf{exhibits}\ (\langle\rangle, \{\})}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{Q} \wedge \mathtt{Q}\ \textbf{exhibits}\ (s, X)}{P\ \textbf{exhibits}\ (\langle(0, a)\rangle^\frown s, X)} \quad [\ a \neq i\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{i}} \mathtt{Q} \wedge \mathtt{Q}\ \textbf{exhibits}\ (s, X)}{\mathtt{P}\ \textbf{exhibits}\ (s, X)}$$

$$\frac{\begin{array}{l}\mathtt{P} \xrightarrow{\mathtt{t}} \mathtt{Q} \\ \mathtt{Q}\ \textbf{exhibits}\ (s, X - t) \\ \forall\, a \in \sigma(X \lhd t) \bullet \neg(\mathtt{P} \xrightarrow{\mathtt{a}}\ )\end{array}}{\mathtt{P}\ \textbf{exhibits}\ (s + t, X)}$$

The relation **exhibits** is defined to be the smallest relation closed under these inference rules—it holds only in cases that can be established using these rules.

The last consistency rule exploits a variant of the reverse persistency result 4.2.4 of [LeL94]: that if $\mathtt{P} \xrightarrow{\mathtt{t}} \mathtt{Q}$ and $\neg(\mathtt{P} \xrightarrow{\mathtt{a}}\ )$ then

$$\forall\, t' < t \bullet \mathtt{P} \xrightarrow{\mathtt{t'}} \mathtt{P'} \Rightarrow \neg(\mathtt{P'} \xrightarrow{\mathtt{a}}\ )$$

In other words, if `P` is able to refuse `a` initially, then at every point before the end of an evolution `a` may still be refused. Hence the refusal set $X \lhd t$ recorded during the evolution is consistent with the execution if `P` is unable to perform any of the events in that set (at the beginning of that evolution).

The failures of `P` can now be extracted:

$$\mathit{failures}(\mathtt{P}) \;=\; \{(s, X) \mid \mathtt{P}\ \textbf{exhibits}\ (s, X)\}$$

Example

The following sequence of executions is possible for $\mathtt{a}\,;\Delta^2\,\mathtt{b}\,;\mathtt{stop}$, and this is consistent with the failure $(\langle(3, a), (8, b)\rangle, [0, 3) \times \{b\} \cup [3, 5) \times \{a\})$.

$$\begin{tikzcd}[row sep=small]
\mathit{0} & \mathtt{a}\,;\Delta^2\,\mathtt{b}\,;\mathtt{stop} \arrow[d, "\mathit{3}"] & & & \\
\mathit{3} & \mathtt{a}\,;\Delta^2\,\mathtt{b}\,;\mathtt{stop} \arrow[r, "a"] & \Delta^2\,\mathtt{b}\,;\mathtt{stop} \arrow[d, "\mathit{2}"] & & \\
5 & & \Delta^0\ ;\mathtt{b}\,;\mathtt{stop} \arrow[d, "\mathit{3}"] & & \\
\mathit{8} & & \mathtt{b}\,;\mathtt{stop} \arrow[r, "b"] & \mathtt{stop} &
\end{tikzcd}$$

The inference rules may be used with this sequence of transitions (considered in reverse order) to derive the claim above:

$$\begin{array}{lrll}
 & \texttt{stop} & \textbf{exhibits} & (\langle\rangle, \{\}) \\
\vdash & \texttt{b}\,;\texttt{stop} & \textbf{exhibits} & (\langle(0, b)\rangle, \{\}) \\
\vdash & \Delta^0\,\texttt{b}\,;\texttt{stop} & \textbf{exhibits} & (\langle(3, b)\rangle, \{\}) \\
\vdash & \Delta^2\,\texttt{b}\,;\texttt{stop} & \textbf{exhibits} & (\langle(5, b)\rangle, [0, 2) \times \{a\}) \\
\vdash & \texttt{a}\,;\Delta^2\,\texttt{b}\,;\texttt{stop} & \textbf{exhibits} & (\langle(0, a), (5, b)\rangle, [0, 2) \times \{a\}) \\
\vdash & \texttt{a}\,;\Delta^2\,\texttt{b}\,;\texttt{stop} & \textbf{exhibits} & (\langle(3, a), (8, b)\rangle, [0, 3) \times \{b\} \cup [3, 5) \times \{a\})
\end{array}$$

The traces of a process may be deduced from its failures:

Theorem 1.1 $traces(P) = \{\text{events}(s) \mid (s, X) \in failures(P)\}$ □

Proof Each execution of P gives rise to a trace tr, and is consistent with a set of failures of P each of whose timed trace s has $\text{events}(s) = tr$. □

Corollary 1.2 Timed failures equivalence is stronger than trace equivalence □

Distinguishing processes with different failures

The next theorem gives the converse relationship between trace congruence and timed failures equivalence. If failures equivalence were a congruence, then the previous corollary together with this theorem would yield that failures equivalence is trace congruence. Alas, as we shall see later, failures equivalence is not a congruence for the language Basic ET-LOTOS.

Theorem 1.3 If two processes are trace congruent, then they have the same timed failures □

Proof Given a particular timed failure (s_0, X_0), with a particular representation for the refusal set X_0

$$X_0 = [u_1, u_1') \times A_1 \cup [u_2, u_2') \times A_2 \cup \ldots \cup [u_m, u_m') \times A_m$$

we can construct a context that detects (s_0, X_0).

$$\begin{array}{rcl}
\mathtt{P}'_{\langle\rangle} & = & \omega\,;\texttt{Stop} \\
\mathtt{P}'_{\langle(\mathtt{t},\mathtt{a})\rangle\frown\mathtt{s}} & = & \Delta^{\mathtt{t}}\,\mathtt{a@t\{0\}}\,;\mathtt{P}'_{\mathtt{s-t}} \\
\mathtt{Q}'_{\mathtt{i}} & = & []_{\mathtt{a}\in\mathtt{A_i}}\Delta^{\mathtt{u}}\,\mathtt{a@t\{u'-u\}}\,;\texttt{Stop} \;\Box\; \Delta^{\mathtt{u}'}\,\mathtt{i}\,;\omega\,;\texttt{Stop} \\
\mathtt{Q}' & = & \big\|_{\omega}\,\mathtt{Q}'_{\mathtt{i}}
\end{array}$$

In this construction, ω is a special 'success' event that is not contained in the set of gates G available to processes. Note the use of generalised choice in the definition of $\mathtt{Q_i}$; since the set A_i is finite, this is simply multiple applications of the $\Box$ operator.

The timed LOTOS context for (s_0, X_0) is

$$\mathtt{C}_{(\mathtt{s_0},\mathtt{X_0})}(\mathtt{R}) \;=\; (\mathtt{R}\,|[\,\Sigma\,]|\,(\mathtt{P}'_{\mathtt{s_0}}\,|[\,\omega\,]|\,\mathtt{Q}'_{\mathtt{X_0}}))\ \backslash\ \Sigma$$

The process $\mathtt{C}_{(\mathtt{s_0},\mathtt{X_0})}(\mathtt{P})$ is able to perform ω if and only if $(s_0, X_0) \in \mathit{failures}(\mathtt{P})$: the $\mathtt{P}'_{\mathtt{s_0}}$ component can perform ω only after performing the events in the trace s_0, and the $\mathtt{Q}'_{\mathtt{X_0}}$ component can perform ω only if everything in X_0 is refused.

If (s_0, X_0) is a failure of $\mathtt{P}$, then $\mathit{traces}(\mathtt{C}_{(\mathtt{s_0},\mathtt{X_0})}(\mathtt{P})) = \{\langle\rangle, \langle\omega\rangle\}$. If (s_0, X_0) is not a failure of $\mathtt{P}$, then $\mathit{traces}(\mathtt{C}_{(\mathtt{s_0},\mathtt{X_0})}(\mathtt{P})) = \{\langle\rangle\}$. Thus if two processes have the same traces in all contexts, then they must have the same timed failures. $\Box$

This result also holds for timed CSP. There it is also true that failures equivalence is a congruence, and so it follows that it is trace congruence for the language of timed CSP.

Failures equivalence is not a congruence

In the case of ET-LOTOS, however, failures equivalence is not enough to yield a congruence. Failures contain no information about internal activity. This does not matter in timed CSP, since none of the operators of the language respond to internal events. In ET-LOTOS, there are two operators that are sensitive to hiding, and which therefore may not preserve failures equivalence: choice and disabling.

Consider the two processes

$$\begin{aligned} \mathtt{P1} &= \mathtt{i}\,;\mathtt{a}\{5\}\,;\mathtt{Stop} \\ \mathtt{P2} &= \mathtt{a}\{5\}\,;\mathtt{Stop} \end{aligned}$$

These two processes have the same timed failures:

$$\begin{aligned} &\{(\langle\rangle, X) \mid a \notin \sigma(X \uparrow [0,5])\} \\ &\cup\{(\langle(u,a)\rangle, X) \mid u \le 5 \wedge a \notin \sigma(X \uparrow [0,u))\} \end{aligned}$$

But $\mathtt{P_1}$ will always perform an internal event at time 0, whereas $\mathtt{P_2}$ performs no event before its initial $\mathtt{a}$.

A context $\mathtt{C(R)} = \mathtt{R}\ \Box\ \Delta^1\,\mathtt{b}\{5\}\,;\mathtt{Stop}$, involving choice may distinguish $\mathtt{P1}$ from $\mathtt{P2}$: the process $\mathtt{C(P1)}$ is unable to perform a $\mathtt{b}$ event at time 1—the internal event of $\mathtt{P1}$ will resolve the choice at time 0. On the other hand, the process $\mathtt{C(P2)}$ is able to perform event $\mathtt{b}$ at time 1. Thus $\langle b\rangle$ is a trace of $\mathtt{C(P2)}$ but not of $\mathtt{C(P1)}$.

A context $\mathtt{D(R)} = \Delta^1 \,\mathtt{b\{5\}\,;Stop\ [>\ R}$ involving disabling may distinguish $\mathtt{P_1}$ from $\mathtt{P_2}$: `D(P1)` is unable to perform `b` since the interrupt will occur at time *0*. Yet `D(P2)` is able to perform `b`, since the interrupt need not occur at all, and certainly need not occur before time *1* when `b` becomes available.

These examples illustrate that the timed failures information is not sufficient to predict all the traces of a process in any context.

The extra information required is the time of the first event (internal or external) during a particular execution corresponding to a timed failure. This will be the time at which a choice is resolved, or at which a disabling occurs. It is only the first event that can be detected by these two operators, so information concerning subsequent internal activity is not required.

We will associate a time as well as a timed failure with a given execution. A number of time values could be associated with any particular timed failure. A timed failure triple is a triple:

$$(t, s, X)$$

which is associated with a process iff the process has an execution where the first (internal or external) event occurs at time t, s records the trace of visible events, and X records the visible events that were refused. If $s = \langle\rangle$ then t can take a finite or infinite value; otherwise it must be finite (in fact no greater than the beginning of s). The infinite value in $(\infty, \langle\rangle, X)$ indicates that there is an execution consistent with $(\langle\rangle, X)$ in which no event occurs—in other words, no event need occur before $end(X)$.

These triples can be extracted directly from the operational semantics in much the same way as the timed failures were earlier:

$$\frac{}{\mathtt{P}\ \mathbf{exhibits}'\ (\infty, \langle\rangle, \{\})}$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{a}} \mathtt{Q} \qquad \mathtt{Q}\ \mathbf{exhibits}'\ (t, s, X)}{\mathtt{P}\ \mathbf{exhibits}'\ (\mathit{0}, \langle(\mathit{0}, a)\rangle^\frown s, X)} \quad [\ \mathtt{a} \neq \mathtt{i}\ ]$$

$$\frac{\mathtt{P} \xrightarrow{\mathtt{i}} \mathtt{Q} \qquad \mathtt{Q}\ \mathbf{exhibits}'\ (t, s, X)}{\mathtt{P}\ \mathbf{exhibits}'\ (\mathit{0}, s, X)}$$

$$\frac{\begin{array}{l}\mathtt{P} \xrightarrow{\mathtt{d}} \mathtt{Q} \\ \mathtt{Q}\ \mathbf{exhibits}'\ (u, s, X - d) \\ \forall\, a \in \sigma(X \lhd d) \bullet \neg(\mathtt{P} \xrightarrow{\mathtt{a}}\ )\end{array}}{\mathtt{P}\ \mathbf{exhibits}'\ (u + t, s + t, X)}$$

We associate a set of failure triples with a process P

$$\mathit{failuretriples}(\mathtt{P}) \;=\; \{(t, s, X) \mid \mathtt{P}\ \mathbf{exhibits}'\ (t, s, X)\}$$

The following theorem is immediate:

Theorem 1.4 $\mathit{traces}(\mathtt{P}) = \{\mathrm{events}(s) \mid (t, s, X) \in \mathit{failuretriples}(\mathtt{P})\}$ □

Proof Each execution of P gives rise to a trace tr, and is consistent with a set of failure triples of P each of whose timed trace s has $\mathrm{events}(s) = tr$. □

Corollary 1.5 Failure triple equivalence is stronger than trace equivalence □

The next theorem points in the opposite direction.

Theorem 1.6 Trace congruence is stronger than failure triple equivalence □

Proof We have two cases to consider: t finite and t infinite.

If t is finite then let $\mathtt{C}_{(\mathtt{t},\mathtt{s},\mathtt{X})}(\mathtt{R})$ be the context

$$\mathtt{C}_{\langle(\mathtt{t},\upsilon)\rangle^\frown \mathtt{s},\mathtt{X}\cup[\mathtt{t},\mathtt{t+1})\times\{\upsilon\}}(\upsilon\,;\upsilon\,;\mathtt{Stop}\ \mathtt{[>}\ \mathtt{R})$$

Observe that $t \leq \mathit{begin}(s)$, so $\mathit{trace}(t, \upsilon)^\frown s$ is well-formed. Then for υ to be performed and then refused at time t, we must have that the disabling occurs at time t: the first event from the execution of R which corresponds to (s, X). Hence $\mathtt{C}_{(\mathtt{t},\mathtt{s},\mathtt{X})}(\mathtt{P})$ will have $\langle\omega\rangle$ as a trace if and only if (t, s, X) is a failure triple of P.

Now consider the case that t is infinite. We first require a lemma:

Lemma 1.7 If P $\mathbf{exhibits}'$ $(t, \langle\rangle, X)$ and $t \geq \mathit{end}(X)$ then P $\mathbf{exhibits}'$ $(\infty, \langle\rangle, X)$ □

This is easily established by examining the proof of $\mathtt{P}$ **exhibits**$'$ $(t, \langle\rangle, X)$, and considering the prefix of it which has inferences based only on delay transitions.

Now if $(\infty, \langle\rangle, X)$ is a behaviour of $\mathtt{P}$ but not of $\mathtt{Q}$, then there are two possibilities: either $(\langle\rangle, X)$ is not a failure of $\mathtt{Q}$, in which case there is a context $\mathtt{C}_{(\langle\rangle,\mathtt{X})}$ to distinguish $\mathtt{P}$ from $\mathtt{Q}$; or else it is a failure of $\mathtt{Q}$, in which case any associated time value must be less than $end(X)$, by the lemma above. In this latter case, consider the context

$$\mathtt{D(R)} \;=\; \mathtt{C}_{(\langle(\mathtt{end(X)},\upsilon)\rangle,\mathtt{X})}(\mathtt{R} \;[]\; \Delta^{\mathtt{end(X)}}\, \upsilon\{\mathtt{0}\}\,;\mathtt{Stop}$$

Then $\mathtt{D(P)}$ can perform an ω, since it is possible that the choice will not be resolved by $\mathtt{P}$ by time $end(X)$. On the other hand, by the lemma above, any execution of $\mathtt{Q}$ exhibiting $(\langle\rangle, X)$ must perform some event strictly earlier than $end(X)$, and so the choice can never be resolved against $\mathtt{Q}$ during such an execution. Thus $\mathtt{D(Q)}$ cannot perform ω. So there is a context that distinguishes $\mathtt{P}$ from $\mathtt{Q}$.

We conclude that trace congruence implies failure triple equivalence; processes with different timed failure triples can be placed in contexts that yield different traces.
□

Now we can show that failure triple equivalence is a congruence. It will then follow from Theorem 1.6 and Corollary 1.5 that failure triple equivalence is trace congruence.

Theorem 1.8 Failure triple equivalence is a congruence □

Proof We show this by giving the semantic equations for timed LOTOS in terms of timed failure triples. We must prove that these equations correspond to the result of extracting the timed failure triples directly from the operational semantics. Since the equations are given in a compositional way, they automatically define a congruence.

We establish the following by structural induction on $\mathtt{P}$, where is the semantic function from ET-LOTOS to the semantic domain TM_L defined below:

$$\forall\, \mathtt{P} \bullet \mathit{failuretriples}(\mathtt{P}) \;=\; [\![\mathtt{P}]\!]$$

This completes the proof. □

Denotational semantics for ET-LOTOS

The semantic domain TM_L is defined to be those sets

$$S \subseteq ([\mathit{0}, \infty) \times \mathit{TT} \times \mathit{FRSET}) \cup (\{\infty\} \times \{\langle\rangle\} \times \mathit{FRSET})$$

of failure triples satisfying the following condition:

$$(C)\ (t, \langle\rangle, X) \in S \wedge t \geq end(X) \Rightarrow (\infty, \langle\rangle, X) \in S$$

The semantic function is a mapping ET-LOTOS $\rightarrow$ TM_L, defined by the following equations:

$$\begin{array}{rcl}
[\![\texttt{stop}]\!] & = & \{(\infty, \langle\rangle, X) \mid X \in RSET\} \\
[\![\texttt{exit\{d\}}]\!] & = & \{(\infty, \langle\rangle, X) \mid ([0, d] \times \{\delta\}) \cap X = \{\}\} \\
 & & \cup \\
 & & \{(t, \langle(t, \delta)\rangle, X) \mid ([0, t) \times \{\delta\}) \cap X = \{\} \wedge t \leq d\} \\
[\![\texttt{a@t\{d\};Q}]\!] & = & \{(\infty, \langle\rangle, X) \mid ([0, d] \times \{a\}) \cap X = \{\}\} \\
 & & \cup \\
 & & \{(u, \langle(u, a)\rangle ^\frown (s + u), X) \mid \ ([0, u) \times \{a\}) \cap X = \{\} \wedge u \leq d \\
 & & \qquad\qquad\qquad\qquad\qquad \wedge \exists\, v \bullet (v, s, X - u) \in [\![\texttt{Q[u/t]}]\!]\} \\
[\![\texttt{i@t\{d\};Q}]\!] & = & \{(\infty, \langle\rangle, X) \mid end(X) \leq d\} \\
 & & \cup \\
 & & \{(u, s + u, X) \mid u \leq d \wedge \exists\, v \bullet (v, s, X - u) \in [\![\texttt{Q[u/t]}]\!]\} \\
[\![\Delta^{\texttt{d}}\,\texttt{Q}]\!] & = & \{(u + d, s + d, X) \mid (u, s, X - d) \in [\![\texttt{Q}]\!]\} \\
[\![\texttt{P}\ \Box\ \texttt{Q}]\!] & = & \{(u, s, X) \mid \ (u, s, X) \in [\![\texttt{P}]\!] \wedge \\
 & & \qquad\qquad\quad \exists\, v \geq u \bullet (v, \langle\rangle, X \lhd u) \in [\![\texttt{Q}]\!]\} \\
 & & \cup \\
 & & \{(u, s, X) \mid \ (u, s, X) \in [\![\texttt{Q}]\!] \wedge \\
 & & \qquad\qquad\quad \exists\, v \geq u \bullet (v, \langle\rangle, X \lhd u) \in [\![\texttt{P}]\!]\} \\
[\![\texttt{P}\,|[\Gamma]|\,\texttt{Q}]\!] & = & \{(u, s, X) \mid \ \exists (u_P, s_P, X_P) \in [\![\texttt{P}]\!], (u_Q, s_Q, X_Q) \in [\![\texttt{Q}]\!] \bullet \\
 & & \qquad\qquad\quad u = min\{u_P, u_Q\} \wedge s \in s_P\,|[\Gamma]|\,s_Q \\
 & & \qquad\qquad\quad \wedge X \upharpoonright \Gamma = X_P \upharpoonright \Gamma \cup X_Q \upharpoonright \Gamma \\
 & & \qquad\qquad\quad \wedge X \setminus \Gamma = X_P \setminus \Gamma \cap X_Q \setminus \Gamma\} \\
[\![\texttt{hide}\ \Gamma\ \texttt{in Q}]\!] & = & \{(u, s \setminus \Gamma, X) \mid (u, s, X \cup [0, end(s, X)) \times \Gamma) \in [\![\texttt{Q}]\!]\} \\
[\![\texttt{P} \gg \texttt{Q}]\!] & = & \{(u, s, X) \mid \ (u, s, X \cup [0, end(s, X)) \times \{\delta\}) \in [\![\texttt{P}]\!] \wedge \delta \notin \sigma(s)\} \\
 & & \cup \\
 & & \{(u, s ^\frown (s' + t), X) \mid \ (u, s ^\frown \langle(t, \delta)\rangle, X \lhd t \cup [0, t) \times \{\delta\} \in [\![\texttt{P}]\!] \\
 & & \qquad\qquad\qquad\qquad\quad \wedge \delta \notin \sigma(s) \\
 & & \qquad\qquad\qquad\qquad\quad \wedge \exists\, v \bullet (v, s', X - t) \in [\![\texttt{Q}]\!]\}
\end{array}$$

$$
\begin{aligned}
[\![\texttt{P [> Q}]\!] \;=\; & \{(u,s,X) \mid (u,s,X) \in [\![\texttt{P}]\!] \wedge \\
& \qquad\qquad \exists\, v \geq u, v \geq end(s,X) \bullet (v, \langle\rangle, X) \in [\![\texttt{Q}]\!]\} \\
& \cup \\
& \{(u, s^\frown s', X) \mid \exists (u_P, s, X_P) \in [\![\texttt{P}]\!], (u_Q, s', X) \in [\![\texttt{Q}]\!] \bullet \\
& \qquad\qquad X_P = X \lhd u_Q \wedge end(s) \leq u_Q \wedge u = min(u_P, u_Q)\}
\end{aligned}
$$

To give a semantics to recursively defined processes, we impose a metric space structure on TM_L. Define

$$
\begin{aligned}
S \lhd t \;&=\; \{(u,s,X) \mid (u,s,X) \in S, u < t, end(s,X) < t\} \\
&\quad \cup \\
&\quad \{(\infty, \langle\rangle, X) \mid (t', \langle\rangle, X) \in S, end(X) < t \leq t'\} \\
d(S_1, S_2) \;&=\; \inf\{2^{-t} \mid S_1 \lhd t = S_2 \lhd t\}
\end{aligned}
$$

Theorem 1.9 The distance function d is a metric on TM_L, and (d, TM_L) is a complete metric space. □

The body of a recursive definition `Y := Q` is a function from `Y` to `Q` which we require to be time guarded. The mapping on TM_L corresponding to this function is a contraction mapping. By Banach's fixed point theorem [Sut75] this mapping has a unique fixed point, since TM_L is a complete metric space. Full details of this argument are presented in [DaS93].

The semantics of `P := Y` where `Y` $=$ `Q` is defined to be the unique fixed point of the contraction mapping corresponding to `Q`.

A note on the structure of timed refusals

Timed refusals are carefully defined to be left-closed and right-open, so refusal information has the form $[b, e) \times A$. This decision reflects the way in which processes make events available. For a context to discover that events are being refused by a process, the context must itself offer them, and the transition system defined for Basic ET-LOTOS has the property that events are always made available at some first point in time, so we may as well record refusal information as being closed at the bottom. In fact, if we allow refusal information based on open intervals as well, then we may end up distinguishing processes that cannot be told apart in any context. Consider the following:

```
P1  :=  (i ; a{0} ; stop □ i ; b ; stop □ i ; b{0} ; stop)
P2  :=  i ; ((a{0} ; stop |[a]| i{∞} ; a ; stop) □ b ; stop)
```

Each of these processes performs its first event (internal) at time 0. Each can perform `a` at time 0 only, and can perform `b` at any time. Each can refuse `b` at time 0 provided `a` is not refused at that time; and each can refuse `b` at any time after time 0. In fact, they have the same failure triples, and no context can distinguish them. However, if we were to allow refusal information also in terms of open intervals, then we would expect $(0, \langle\rangle, [0, 1) \times \{a\} \cup (0, 1) \times \{b\})$ to be exhibited by the `P1`, but not by `P2`, which can exhibit only $(0, \langle\rangle, [0, 1) \times \{a\} \cup [\epsilon, 1) \times \{b\})$ for arbitrarily small $\epsilon > 0$. Thus we would distinguish processes that no context can distinguish.

The reason for using right-open intervals is more pragmatic: no new information would be gained by including right-closed intervals, again because of the property that events are always enabled at first instants. If a refusal $[b, e] \times A$ is possible for a process, then there will always be some $\epsilon > 0$ for which $[b, e + \epsilon) \times A$ is also a refusal; the events in A will not be enabled until some positive time after e. Hence including refusal information based on closed intervals yields no new information about a process.

If events need not become enabled at particular instants, but may become enabled strictly after some instant, then we will need to record other kinds of refusal information. In Full ET-LOTOS, a selection predicate is permitted to restrict the times at which events may occur, so for example the construction `b @ t[t > 0] ; stop` allows `b` to occur at any time after time 0; there is no moment at which it becomes available. In fact, depending on the freedom one is allowed in the predicate, intervals may not be suitable at all (for example, if `a` is permitted to occur only at a rational time) and it seems that the appropriate refusal sets will have significantly less structure, perhaps ranging over arbitrary sets of timed events. In any case, contexts can now be constructed that make finer distinctions. For example, the context for `Y`

$$\begin{array}{l}\texttt{hide } \{\texttt{a,b}\} \texttt{ in } ((\texttt{a ; stop} \ \Box \ \texttt{b @ t[t} > \texttt{0] ; stop} \ \Box \ \Delta^2 \omega \texttt{ ; stop}) \ \texttt{|[a,b]|} \ \texttt{Y}) \\ \texttt{|[}\{\}\texttt{]|} \ \Delta^1 i \texttt{ ; stop}\end{array}$$

can distinguish `P1` (which will allow an ω to be performed) from `P2` (which will not).

Comparison with weak bisimulation

Weak timed bisimulation is an equivalence relation defined on processes to abstract away from internal events. The definition in [LeL94] is as follows:

$$P \xrightarrow{d}_* Q \quad \mathit{iff} \quad P \xrightarrow{d1} \xrightarrow{d2} \ldots \xrightarrow{dn} Q \qquad \text{where } d = \Sigma_{i=1}^{n} di$$

$$P \overset{d}{\Longrightarrow} Q \quad \mathit{iff} \quad P(\xrightarrow{i})^* \xrightarrow{d1}_* (\xrightarrow{i})^* \ldots \xrightarrow{dn}_* (\xrightarrow{i})^* Q \qquad \text{where } d = \Sigma_{i=1}^{n} di$$

$$P \overset{a}{\Longrightarrow} Q \;\; \textit{iff} \;\; P(\overset{i}{\longrightarrow})^* \overset{a}{\longrightarrow} (\overset{i}{\longrightarrow})^* Q$$
$$P \overset{\varepsilon}{\Longrightarrow} Q \;\; \textit{iff} \;\; P(\overset{i}{\longrightarrow})^* Q$$

Then a relation R is a weak timed bisimulation if and only if whenever $B_1 \; R \; B_2$ then for any α:

1. if $B_1 \overset{\alpha}{\Longrightarrow} B_1'$ then there is some B_2' such that $B_2 \overset{\alpha}{\Longrightarrow} B_2'$ and $B_1' \; R \; B_2'$
2. if $B_2 \overset{\alpha}{\Longrightarrow} B_2'$ then there is some B_1' such that $B_1 \overset{\alpha}{\Longrightarrow} B_1'$ and $B_1' \; R \; B_2'$

Weak timed bisimulation equivalence is incomparable with failure triple congruence. For example, the processes `a ; stop` and `i ; a ; stop` are weak timed bisimulation equivalent, but have different failure triples: for example the failure triple $(1, \langle(1, a)\rangle, \{\})$ is a possible behaviour of the first process, but not of the second (since the time value must be 0 with such a trace).

However, weak timed bisimulation *congruence* will be stronger than failure triple equivalence, as will be illustrated by the following examples, where pairs of processes are exhibited which have the same failure triple sets, but are not bisimilar.

Consider the following pair of processes:

$$\begin{aligned} &\texttt{P1} \;\; \texttt{:=} \;\; (\texttt{i ; } \Delta^3 \, \texttt{a\{5\} ; stop}) \; \Box \; (\texttt{i ; } \Delta^3 \, \texttt{b\{5\} ; stop}) \\ &\texttt{P2} \;\; \texttt{:=} \;\; \texttt{i ; } \Delta^3 \, ((\texttt{i ; a\{5\} ; stop}) \; \Box \; (\texttt{i ; b\{5\} ; stop})) \end{aligned}$$

Each of `P1` and `P2` will nondeterministically offer either an `a` or a `b` over the interval $[3, 8]$, so they have the same failure triple semantics. But `P1` resolves the choice at time 0, whereas `P2` does not resolve the choice until time 3, so it can allow three units of time to pass (performing only its internal transition at time 0) and reach a state from which both a and b are possible; this delay cannot be matched by `P1`.

This next pair of processes also have the same failure triple sets—no context can distinguish them—but they are not bisimilar.

$$\begin{aligned} \texttt{P3} &= \texttt{i ; stop} \; \Box \; \texttt{i ; a ; stop} \\ \texttt{P4} &= \texttt{P3} \; \Box \; \texttt{i ; a\{2\} ; stop} \end{aligned}$$

The process `P3` nondeterministically chooses either to deadlock immediately, or to offer the event `a` until it is accepted. Process `P4` has these same choices, but may also choose to offer `a` for two units of time only. This is a transition that cannot be matched by `P3`.

A standard example from untimed process algebra also illustrates the fact that failures equivalence is weaker than weak timed bisimulation congruence. Consider the following pair of processes:

$$\begin{aligned}\texttt{P5} &= \texttt{i}\,;\texttt{a}\{4\}\,;\texttt{stop}\ \Box\ \texttt{i}\,;\texttt{b}\{4\}\,;\texttt{stop}\\ \texttt{P6} &= \texttt{P5}\ \Box\ \texttt{i}\,;(\texttt{a}\{4\}\,;\texttt{stop}\ \Box\ \texttt{b}\{4\}\,;\texttt{stop}\end{aligned}$$

The first process offers either an `a` or a `b` event for four units of time. The second may offer either of these, or else may offer its environment the choice of both. They are not bisimilar: `P6` can reach a state in which both `a` and `b` are offered, and `P5` can reach no similar state. Again, no context can distinguish these processes.

Urgency

The model TM_L does not handle the phenomenon of 'urgency' or forcing of events, where events must occur by some particular time. Urgent events can be modelled in ET-LOTOS by judicious use of timestops: the process $\texttt{a}\,;\texttt{P}\ \Box\ \texttt{timestop}$ where *timestop* `:=` *timestop* forces the event `a` to occur at time *0*. However, Leduc and Léonard do not encourage the use of timestops, and take the view that timestops are unnecessary in specification.

For the denotational semantics, we were careful to forbid non-time-guarded recursions, and no timestopping processes can be constructed from the operators of Basic ET-LOTOS without such recursive definitions. The handling of forced events and time-stopping processes within a denotational model is a non-trivial extension of the model presented here. In the document [BDS94] an adaptation of an earlier *signals* model is proposed, which includes the modelling of urgent events.

As an example of the sorts of difficulties that arise when internal events can be made urgent, we will examine the urgency operator `urge a in P` of [BLT94], which makes all occurrences of `a` urgent in process `P`. This may be used on internal or external actions.

Consider the two processes below. They have the same failure triples. Each may offer nondeterministically either `a` or `b` sometime between time *1* and time *3*.

$$\begin{aligned}\texttt{P7} &:= \texttt{i}\,;(\Delta^1\ ;\texttt{i}\{2\}\,;\texttt{a}\,;\texttt{stop}\ \Box\ \texttt{i}\{2\}\,;\Delta^1\ ;\texttt{b}\,;\texttt{stop})\\ \texttt{P8} &:= \texttt{i}\,;(\Delta^1\ ;\texttt{i}\{2\}\,;\texttt{b}\,;\texttt{stop}\ \Box\ \texttt{i}\{2\}\,;\Delta^1\ ;\texttt{a}\,;\texttt{stop})\end{aligned}$$

If the internal event is made urgent, then the resulting processes are different:

$$\texttt{urge i in P7} \neq \texttt{urge i in P8}$$

The left-hand process can offer only `b`, and the right-hand can offer only `a`. So failure triples is no longer a congruence in the presence of this operator; more information is required in order to obtain a congruence. However, we conjecture that the `urge` operator restricted to external actions will preserve traces-and-timestops congruence over the language of Basic ET-LOTOS.

Acknowledgements

The authors would like to thank the members of the CONCUR2 project for comments on an earlier version of this material presented in Brighton, March 1994, and also the participants of the COST 247 WG1 workshop on extended process algebras, also in Brighton, July 1994.

Steve Schneider is grateful to the CONCUR2 project and the COST 247 project for providing funding towards this research. Jim Davies is grateful to the DRA for funding.

References

[BDS94] J. Bryans, J.W. Davies, and S.A. Schneider, *Denotational semantic models for real-time LOTOS,* Brighton COST 247 WG1 workshop on extended process algebras, July 1994.

[BLT94] T. Bolognesi, F. Lucidi and S. Trigila, *Converging towards a Timed LOTOS Standard,* Computer Standards and Interfaces, Vol. 16 1994.

[DaS93] J.W. Davies and S.A. Schneider, *Recursion induction for real-time processes,* Formal Aspects of Computing, 5(6), 1993.

[Hen88] M. Hennessy, *Algebraic theory of processes,* MIT Press, 1988.

[LeL94] G. Leduc and L. Leonard, *A Formal Definition of Time in LOTOS,* University of Liege, 1994.

[QFA93] J. Quemada, D. Frutos and A. Azcorra, *TIC: A TImed Calculus,* Formal Aspects of Computing, 1993.

[Sch94] S.A. Schneider, *An operational semantics for timed* CSP Information and Computation, to appear, 1994

[Sut75] W. A. Sutherland, *Introduction to Metric and Topological Spaces*, Oxford University Press 1975.