An Extended LOTOS for the design of Real-Time Systems

Luc Léonard and Guy Leduc
Research Assistant and Research Associate of the National Fund for Scientific Research (Belgium)
Université de Liège, Institut d’Electricité Montefiore, B 28, B-4000 Liège 1, Belgium
Tel: + 32 4 3662697       Fax: + 32 4 3662989     E-mail: leonard@montefiore.ulg.ac.be

1. Introduction

We give in the following a brief presentation of ET-LOTOS [LéL 95a, LéL 95b]. ET-LOTOS extends with quantative time the formal description technique LOTOS [ISO 8807]. Other proposals for a “time extended” LOTOS exist. Let us mention [QMF 94] and [BLT 94]. ET-LOTOS serves as basis for the time extension part of E-LOTOS, the new standard for LOTOS currently developed within ISO (ISO/IEC JTC1/SC21).

We assume in the sequel that the reader has a basic knowledge of the syntax and the semantics of LOTOS.

2. Formal semantics and properties of ET-LOTOS

2.1. Datatypes and time domain

In ET-LOTOS, like in LOTOS, datatypes are described in the Abstract Datatype language ACT ONE, that has an initial semantics.

The time domain, denoted \( \mathbb{D} \), is defined as the set of values of a given data sort \( \text{time} (\mathbb{D} = \mathbb{Q} (\text{time})) \). Its definition is left free to the will of the specifier provided that the following elements be defined.

- A total order relation represented by ">".
- An element \( 0 \in \mathbb{D} \) such that: \( \forall r \in \mathbb{D}: r\neq 0 \Rightarrow r>0 \)
- An element \( \infty \in \mathbb{D} \) such that: \( \forall r \in \mathbb{D}: r\neq \infty \Rightarrow \infty>r \)
- A commutative and associative operation \( + : \mathbb{D}, \mathbb{D} \rightarrow \mathbb{D} \) such that:
  \[ \forall r, r_1 \in \mathbb{D}: r>r_1 \Leftrightarrow \exists r'>0 \cdot (r'+r_1)=r \]
  \[ \forall r, r_1 \in \mathbb{D}: r>0 \text{ and } r_1\neq \infty \Rightarrow r+r_1\in \mathbb{D} \]
  \[ \forall r \in \mathbb{D}: r+0=r \]
  \[ \forall r \in \mathbb{D}: r+\infty=\infty \]

The relations “\( \leq \)”, and “\( - \)” can be derived easily as follows:

\[ \forall r, r_1 \in \mathbb{D} \cdot r \leq r_1 \Leftrightarrow (r < r_1 \lor r_1 = r) \]
\[ \forall r, r_1, r_2 \in \mathbb{D} \cdot r \leq r_2 \Rightarrow (r - r_1 \leq r_2 \Leftrightarrow r_2 + r_1 = r) \]
\[ \forall r, r_1 \in \mathbb{D} \cdot r \leq r_1 \Rightarrow r - r_1 = 0 \]
In particular, the time domain can be dense as well as discrete, but to be able to give the operational semantics of ET-LOTOS in terms of Labelled Transition Systems (LTS), it must be countable, such as the rational numbers.

### 2.2 Notations

The following notations hold for the remainder of the paper. $\text{G}$ denotes the countable set of common observable gates. $\text{L} = \text{G} \cup \{\delta\}$ denotes the alphabet of observable gates where $\delta$ is the special action denoting successful termination ($\delta \notin \text{G}$). $\delta$ does not appear explicitly in the syntax of LOTOS. $\text{S}$ denotes the set of sorts, $\text{V}$ denotes the set of ground terms in the initial algebra associated with the ACT ONE specification: $\text{V} = \cup_{\text{s}} \text{Q(s)}$. $\text{CL} = \text{L} \times \text{V}^*$ denotes the set of observable actions. $\text{A} = \text{CL} \cup \{i\}$ denotes the alphabet of actions, where the symbol $i$ is reserved for the unobservable internal action ($i \notin \text{L}$). $g$ (resp. $a$) denotes an element of $\text{G}$ (resp. $\text{A}$): $g \in \text{G}$, $a \in \text{A}$. $g\text{v}_1…\text{v}_n$ and $\delta \text{v}_1…\text{v}_n$ denote elements of $\text{CL}$, with the $\text{v}_i$'s in $\text{V}$. Capital Greek letters such as $\Gamma$ will be used to denote subsets of $\text{G}$. $\text{D}$ denotes the countable time domain which is the alphabet of time actions. $D_{0;\infty} = \text{D} - \{0,\infty\}$.

### 2.3 Syntax of the behaviour part of ET-LOTOS

The collection of ET-LOTOS behaviour expressions is defined by the following BNF expressions. In these expressions, $\tilde{x}$ represents a vector of process names, $\text{SP}$ is a selection predicate, the $e_i$'s represent a term $tx$, the $\text{o}_i$'s represent either $?x:s$ (with $x$ a variable of sort $s$) or $!tx$ (with $tx$ a ground term), the $x_i$'s (resp. $tx_i$'s) are variables (resp. ground terms) of sorts $s_i$'s, $d \in \text{D}$ and in $\text{et}$, $t$ is a variable of sort $\text{time}$. The new features are printed in italics:

$$P ::= \text{Q where } X := \text{Q} \quad 2$$
$$\text{Q ::= stop | exit(e_1,…,e_n){d}} | g_1…g_n\text{et}[\text{SP};\text{Q}] | i\text{et}(d);\text{Q} | A^d\text{Q} | \text{Q}[\Gamma] | \text{Q}[\text{SP}] | \text{Q} \mid \text{Q} \mid \text{Q} \mid \text{Q} \mid \text{Q}$$

Remark: in $g_1…g_n\text{et}[\text{SP};\text{Q}]$ we let both $\text{et}$ and $[\text{SP}]$ be optional, and use the convention that, if omitted, $[\text{SP}] = [\text{true}]$. In $i\text{et}(d);\text{Q}$, both $\text{et}$ and $\{d\}$ are optional. If omitted, $d = 0$. Similarly $\{d\}$ is optional in $\text{exit}(d)$, and $\text{exit}$ means implicitly $\text{exit}(\infty)$.

The binding powers of the operators are like in LOTOS. For the new operators, $A^d$ has the same power as action-prefix and $\text{inf} \mid \text{inf}$ the same as $\text{choice} x_1:s_1,…x_n:s_n [{}]$.

An additional shorthand notation: We define the notation $g_1…g_n\{d\};\text{Q}$, for $g_1…g_n\text{et}[\text{t}=\text{d};\text{Q}]$, provided that $t$ be fresh in $\text{Q}$. Under the same restriction, we also introduce the notation

---

1 This term can be: ‘any $s$’ (with $s \in \text{S}$)
2 For convenience, we suppose, without lack of generality, that there is a single where-clause that gathers all the process declarations of the specification.
An Extended LOTOS for the design of Real-Time Systems

g0_1 o_n(d_1,d_2) ; P to mean g[t[d_1 s d_2]] ; P. The meaning of these rewritings will become clear in the next section.

2.4 Semantics of ET-LOTOS

The operational semantics of ET-LOTOS, presented in the following, is of the so-called "time/actions" type. This means that the occurrence of actions and the passing of time are considered as separate concerns, each one being described by a dedicated set of rules.

2.4.1 Notations

P, P', Q, Q' denote ET-LOTOS behaviour expressions.

P dP' a\! A means that process P may engage in action a and, after doing so, behave like process P'.

P dP' a • P dP' means \( P \) may idle (i.e. not execute any action in \( A \)) during a period of \( d \) units of time and, after doing so, behave like process P'.

P dP' / means \( \neg ( P \downarrow P' ) \) i.e. \( P \) cannot perform an action on gate g.

P dP', with \( d \! \! \! D_0 \$ \), means that process P may idle (i.e. not execute any action in \( A \)) during a period of \( d \) units of time and, after doing so, behave like process P'.

P dP', with \( d \! \! \! D \$ \), means that \( \neg / P' \) • P dP', i.e. \( P \) cannot idle during a period of \( d \) units of time. In these expressions, it is required that \( P \) and \( P' \) be closed, i.e. they do not contain free variables.

2.4.2 Inference rules

In the following inference rules, \( d \! \! \! D_0 \$, \( d_1 \! \! \! D \), \( d' \! \! \! D \$ \), \( g \! \! \! G \) and \( a \! \! \! A \).

We introduce a process, denoted block, which has no axiom and no inference rules. This process cannot perform any action and blocks the progression of time.

Inaction

(S) \( \text{stop} \downarrow \text{stop} \)

Remark that \( \text{stop} \) cannot perform any action but can idle.

Exit

(Ex1) \( \text{exit}(e_1,\ldots,e_n)\{d_1\} \downarrow \delta v_1 \ldots v_n \rightarrow \text{stop} \)

where \( v_i = [t_i] \)

\( v_i \in Q(s_i) = \{ [t] \mid t \text{ is a ground term of sort } s_i \} \)

if \( e_i = t_i \) (a ground term)

if \( e_i = \text{any } s_i \)

(Ex2) \( \text{exit}(e_1,\ldots,e_n)\{d_1+d\} \downarrow \text{exit}(e_1,\ldots,e_n)\{d_1\} \)

(Ex3) \( \text{exit}(e_1,\ldots,e_n)\{d_1\} \downarrow \text{stop} \) (\( d > d_1 \))

The \( \{d_1\} \) attribute is called the life reducer. Its role is to restrict the time period during which the process can terminate successfully: \( \text{exit}\{d_1\} \) can only perform \( \delta \) during the next \( d_1 \) time units. If \( \text{exit}\{d_1\} \) has not performed \( \delta \) yet after \( d_1 \) time units, it is too late and the process turns into \( \text{stop} \) (rule Ex3).
An Extended LOTOS for the design of Real-Time Systems

Observable action-prefix

$$(\text{AP1})\quad g_{o_1\cdots o_n} @ t [SP]; P \xrightarrow{g v_1 \cdots v_n} [v_1/o_1, \ldots, v_m/o_m, 0/t] P$$

if

- \[ v_1 \in Q(s) = \{ w \mid w \text{ is a ground term of sort } s \} \]
- \[ v_i/o_i = v_i/x \]
- \[ v_i/o_i \text{ is void} \]

and where

- \[ v_1 = [w] \quad \text{if } o_1 = !w \]
- \[ v_1 \in Q(s) \quad \text{if } o_1 = ?x:s \]
- \[ v_i/o_i = v_i/x \quad \text{if } o_i = ?x:s \]
- \[ v_i/o_i \text{ is void} \quad \text{if } o_i = !w \]

$$(\text{AP2})\quad g_{o_1\cdots o_n} @ t [SP]; P \xrightarrow{d} g_{o_1\cdots o_n} @ t [[t+d/t] SP]; [t+d/t] P$$

In \( \hat{t} \), \( t \) is a variable of sort \( \text{time} \). This variable is used to measure the delay actions were being offered on \( g \) when one occurred. When an action occurs (rule \( \text{AP1} \)), \( t \) is instantiated. Instantiating \( t \) by \( 0 \) is logical: \( g_{o_1\cdots o_n} @ t [SP]; P \) describes a process at a given instant and the counting of \( t \) starts at that instant. So, \( t \) is still at \( 0 \) if the process immediately does an action on gate \( g \). The way the value of \( t \) is kept up to date if \( g_{o_1\cdots o_n} @ t [SP]; P \) idles is defined by \( \text{AP2} \).

The \( t \) variable can appear in the selection predicate \( SP \), if there is one. The conditions joined with \( \text{AP1} \) express that the only possible instantiations for the attributes of \( g \) are the ones that make \( SP \) true at that instant.

Internal action-prefix

$$(\text{I1})\quad i @ t \{d1\}; P \xrightarrow{i} [0/t] P \quad (\text{I2})\quad i @ t \{d1+d\}; P \xrightarrow{d} i @ t \{d1\}; [t+d/t] P$$

There is no rule like \( \text{Ex3} \) for the internal action-prefix. \( i @ t \{d1\}; P \) cannot idle more than \( d1 \) time units. If it reaches this limit, time is blocked. The only solution left is to accomplish \( i \). This means that, in Timed Extended LOTOS, the occurrence of \( i \) is compulsory. The semantics of \( i @ t \{d1\}; P \) is that \( i \) shall occur during the next \( d1 \) time units. On the other hand, the semantics of \( \text{exit}(d1) \) is that \( d \) may occur within the next \( d1 \) time units.

Delay prefixing

$$(\text{D1})\quad P \xrightarrow{\Delta d} P' \quad (\text{D2})\quad \Delta d + d \xrightarrow{d} \Delta d \ P \quad (\text{D3})\quad P \xrightarrow{\Delta d} P' \xrightarrow{d + d} P'$$

\( \Delta d ; P \) expresses that \( P \) will be delayed by \( d \) time units.

Choice

$$(\text{Ch1})\quad P \xrightarrow{Q} P' \quad (\text{Ch1'})\quad P \xrightarrow{Q} P' \quad (\text{Ch2})\quad P \xrightarrow{P', Q} Q'$$

Remark rule \( \text{Ch2} \): the passing of time does not resolve a choice. Rule \( \text{Ch2} \) also states that both operands evolve in time at the same pace.

---

3 Of course, in a choice context, the occurrence of \( i \) could be prevented by another offered action.
Generalized choice

The semantics of choice $x_1:s_1, \ldots, x_n:s_n[P]$ is defined via an auxiliary operator, denoted $A\text{choice}(d)$ $x_1:s_1, \ldots, x_n:s_n[P] \rightarrow P$, where $d \in D \times$. $A\text{choice}$ stands for $A\text{gedChoice}$. By definition, choice $x_1:s_1, \ldots, x_n:s_n[P] = A\text{choice}(0)$ $x_1:s_1, \ldots, x_n:s_n[P]$.

\[ \text{(GC1)} \]
$$[tx_1/x_1, \ldots, tx_n/x_n]P \overset{A\text{choice}(0)}{\rightarrow} P'$$

\[ \text{(GC2)} \]
$$[tx_1/x_1, \ldots, tx_n/x_n]P \overset{d}{\rightarrow} P'', P'' \overset{A\text{choice}(d)}{\rightarrow} P'$$

where the $tx_i$ are ground terms with $[tx_i] \in Q(s_i)$

\[ \text{(GC3)} \]
$$\forall <tx_1, \ldots, tx_n> \cdot [tx_i] \in Q(s_i), i = 1, \ldots, n$$
$$A\text{choice}(d') x_1:s_1, \ldots, x_n:s_n[P] \overset{d}{\rightarrow} A\text{choice}(d+d') x_1:s_1, \ldots, x_n:s_n[P]$$

Parallel composition

\[ \text{(PC1)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P', Q \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} Q'$$

\[ \text{(PC1')} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P', Q \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} Q'$$

\[ \text{(PC2)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P', Q \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} Q'$$

Infinite parallel composition

\[ \text{(IP1)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P', Q \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} Q'$$

\[ \text{(IP2)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P', Q \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} Q'$$

\[ \text{inf} \parallel | | \parallel P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P', Q \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} Q'$$

$\text{inf} \parallel | | \parallel P$ corresponds to an infinity of occurrences of $P$ evolving in parallel. In ET-LOTOS, such a behaviour cannot be described by a recursive process like $P_s := P \parallel | | P_s$, because unguarded recursions block time (see [LeL 95b]).

Hide

\[ \text{(H1)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P'$$

\[ \text{(H2)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P'$$

\[ \text{(H3)} \]
$$P \overset{d}{\rightarrow} P', \forall g \in \Gamma \cdot (P \overset{g}{\rightarrow} \forall P'' \overset{d'}{\rightarrow} P'' \overset{\text{name}(a) \neq \delta}{\rightarrow} P'' \overset{d'}{\rightarrow} )$$

Rule (H3) expresses the maximal progress principle adopted for ET-LOTOS. This principle states that the hidden events must occur as soon as possible. So, the process can only idle if no hidden action is possible.

Enabling

\[ \text{(En1)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P' \cdot \parallel P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P'$$

\[ \text{(En2)} \]
$$P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P' \cdot \parallel P \overset{\text{name}(a) \in \Gamma \cup \{\delta\}}{\rightarrow} P'$$
The occurrence of $\delta$ is hidden by the enabling operator. According to the maximal progress principle, it must occur as soon as possible.

### Disabling

- **Disabling 1 (Di1)**
  \[
  \frac{P \xrightarrow{\mathbf{a}} P'}{P[\mathbb{Q}] \xrightarrow{\mathbf{d}} P'[\mathbb{Q}]} \quad \text{(name(a) = \delta)}
  \]

- **Disabling 2 (Di2)**
  \[
  \frac{Q \xrightarrow{\mathbf{a}} Q'}{P[\mathbb{Q}] \xrightarrow{\mathbf{d}} Q'}
  \]

### Guard

- **Guard 1 (G1)**
  \[
  \frac{P \xrightarrow{\mathbf{a}} P'}{[\mathbf{SP}] \rightarrow P \xrightarrow{\mathbf{d}} P'} \quad \text{if DS \mid SP}
  \]

- **Guard 2 (G2)**
  \[
  \frac{P \xrightarrow{\mathbf{d}} P'}{[\mathbf{SP}] \rightarrow P \xrightarrow{\mathbf{d}} P'} \quad \text{if DS \mid SP}
  \]

- **Guard 3 (G3)**
  \[
  \frac{P \xrightarrow{\mathbf{d}} \text{stop}}{[\mathbf{SP}] \rightarrow P \xrightarrow{\mathbf{d}} \text{stop}} \quad \text{if \neg DS \mid SP}
  \]

### Let

- **Let 1 (L1)**
  \[
  \frac{[g_1/h_1, \ldots, g_n/h_n] \xrightarrow{\mathbf{a}} P', Q[h_1, \ldots, h_n] := P}{P \xrightarrow{\mathbf{d}} P'} \quad \text{let } x_1 = x_1, \ldots, x_n = x_n \text{ in } P \xrightarrow{\mathbf{d}} P'
  \]

- **Let 2 (L2)**
  \[
  \frac{[g_1/h_1, \ldots, g_n/h_n] \xrightarrow{\mathbf{d}} P', Q[h_1, \ldots, h_n] := P}{P \xrightarrow{\mathbf{d}} P'} \quad \text{let } x_1 = x_1, \ldots, x_n = x_n \text{ in } P \xrightarrow{\mathbf{d}} P'
  \]

### Process instantiation

- **In 1 (IN1)**
  \[
  \frac{[g_1/h_1, \ldots, g_n/h_n] \xrightarrow{\mathbf{a}} P', Q[h_1, \ldots, h_n] := P}{Q[g_1, \ldots, g_n] \xrightarrow{\mathbf{d}} P'}
  \]

- **In 2 (IN2)**
  \[
  \frac{[g_1/h_1, \ldots, g_n/h_n] \xrightarrow{\mathbf{d}} P', Q[h_1, \ldots, h_n] := P}{Q[g_1, \ldots, g_n] \xrightarrow{\mathbf{d}} P'}
  \]

Let us outline some interesting features of the semantic rules defined above:

- The LOTOS rules are kept unchanged.
- The alphabet $\mathbb{A}$ of actions is kept as is (e.g. no additional time stamps in action labels). It is just
  extended with time actions from a separate set $\mathbb{D}$.

### 2.5. Properties

ET-LOTOS exhibits many interesting properties (the proofs can be found in \cite{LeL95b}):

- The operational semantics of ET-LOTOS is consistent.
- Time transitions are deterministic: $\forall P \quad (P \xrightarrow{\mathbf{d}} P' \land P \xrightarrow{\mathbf{d}} P") \Rightarrow P' = P"$.
- Time transitions are closed under the relation $\leq$: $P \xrightarrow{\mathbf{d}} \Rightarrow \forall d' \in \mathbb{D} \cdot P \xrightarrow{\mathbf{d} + d'}$.
  Furthermore, $P \xrightarrow{\mathbf{d}} P' \Rightarrow \forall d' \in \mathbb{D} \cdot \exists d'' \cdot P \xrightarrow{\mathbf{d} + d'} P' \land d = d' + d''$.
- Time transitions are additive: $P \xrightarrow{\mathbf{d}} P'$ and $P' \xrightarrow{\mathbf{d}'} P"$ implies $P \xrightarrow{\mathbf{d} + \mathbf{d}'} P"$.
- Strong bisimulation $\simeq$ is a congruence.
- ET-LOTOS is upward compatible with LOTOS, according to the definition given in \cite{NiS92}, but
  for guarded specifications only.
References


