An agent calculus with simple actions where the enabling and disabling are derived operators

T. Massart

Laboratoire d'Informatique Théorique & Helios-B Group, Université Libre de Bruxelles, CP 230, Bld. du Triomphe, 1050 Brussels, Belgium

Communicated by W.M. Turski
Received 22 May 1991

Abstract


We present here a basic agent calculus which uses simple actions and contains six kinds of operators: the sequence, the choice, the parallel composition, the recursion, and finally the hiding and relabelling. We show that the enabling and disabling constructions, defined in the specification language LOTOS, are derived operators in this calculus. As a consequence, this calculus is a good candidate to be used as a basic calculus for LOTOS.

Keywords: Process algebra, enabling, disabling, theory of computation

1. Introduction

The LOTOS Specification Language [5] is a formal technique used to describe concurrent systems. It possesses a rigorous syntax and a semantics aimed at describing the functional part as well as the data part of concurrent systems, using a Process Algebra model based on CCS [10] and CSP [4] associated with an Abstract Data Type model [3]. The semantics of a LOTOS behaviour expression (called agent here) is an interleaving semantics and is based on the Labelled Transition System model [6] (LTS for short). As suggested by Robin Milner [12] and Ed Brinksma [1], it is interesting to first define a basic calculus that represents the behaviour of systems. Only thereafter, a full specification language with data handling, like LOTOS, should be considered. A full language should contain various facilities to "easily" allow the specification of (complex) systems, whereas the corresponding basic calculus should be as simple as possible to ease the mathematical handling of its expressions.

Simple calculi already exist. Their transition relations may be labelled by simple atomic events (as in CCS [10], CSP [4] and Basic LOTOS [2] or by sets (as in LOTCAL [1]) or multisets (as in SCCS [11]) of (sometimes structured) events. Even if, due to the kind of action they use, SCCS and LOTCAL can model more behaviours, it may be preferable (for sake of simplicity) to use a basic calculus with simple actions. But we can fear that the drawback of such an approach is that a lot of basic operators have to be included in the calculus to allow the mapping between the full language and the basic one; this is the case in Basic LOTOS which contains specialized operators, like the enabiling and the disabling.
In the following section, we define a basic calculus which uses simple actions and only six kinds of operators. These operators are quite standard in the literature, but we show, in Section 3, that the LOTOS enabling and disabling are derived operators in our calculus, while the disabling operator is generally considered as non-derivable [1] in such a calculus with simple actions. As a consequence, this calculus is a good candidate to be used as a basic calculus for LOTOS.

2. The basic calculus

Let us briefly define in this section, the syntax of the calculus we shall use and its operational semantics. The operational (interleaving) semantics is restricted to the set of agents, i.e., the set of expressions without free variables (see [7,9] for a more detailed discussion on this subject).

Assume a finite or countably infinite unstructured set, \( \mathcal{L} \), of labels or observable actions, and a separate element \( \tau \), the internal or unobservable action (\( \tau \not\in \mathcal{L} \)). The set of actions, denoted \( \mathcal{A} \), is defined as \( \mathcal{A} = \mathcal{L} \cup \{ \tau \} \). Let \( a, b, \ldots \) range over \( \mathcal{A} \) and \( S \) range over sets of visible actions. We define relabelling functions \( \Phi \) from \( \mathcal{L} \) to \( \mathcal{L} \), and extend \( \Phi \) to \( \mathcal{A} \) by decreeing that \( \Phi(\tau) = \tau \). Assume also a countably infinite set \( \mathcal{X} \) of agent variables, also called variables, ranged over by \( X, Y, \ldots \); the set \( \mathcal{E} \) of agent expressions, also called expressions and ranged over by \( E, E_1, E_2, F, F_1, \ldots \), is defined as the smallest set which contains the following expressions:

\[
\begin{align*}
X & \quad \text{(variable)} \quad | \\
E & \quad \text{(prefix)} \quad | \\
\sum E & \quad \text{(summation)} \quad | \\
E_1 \cdot E_2 & \quad \text{(composition)} \quad |
\end{align*}
\]

where \( X \) is a set of agent variables, and \( E = \langle E_x \rangle_{x \in X} \) is an \( X \)-indexed family of agent expressions in \( \mathcal{E} \).

The operators are given in the order of decreasing priority. If \( E \) is empty, \( \sum E \) is denoted \( 0 \). If \( E \) is finite and \( |E| \geq 2 \), e.g., \( E = \{ E_i \mid 0 \leq i \leq n \} \), for convenience, instead of \( \sum E \) we may write \( E_1 + E_2 + \cdots + E_n \), with the \( E_i \)'s written in any order since no order is given for the elements of the family \( E \).

We can also consider \( X \) as an \( I \)-indexed family of distinct variables and \( E \) as an \( I \)-indexed family of expressions, for some indexing set \( I \), and denote \( \text{fix}_X E = E \) instead of \( \text{fix}_X X = E \). This alternate notion will generally be used in the following. Similarly the family \( E \) in \( \sum E \) may be indexed by some indexing set \( I \).

In \( \text{fix}_X E = E \), the variables \( Y \subset X \) occurring in any \( E \in E \) are bound by the fix operator. For an expression \( E \), \( \text{Vars}(E) \) is the set of free (i.e., not bound) variables of \( E \). An agent expression \( E \) is an agent if it contains no free variables, i.e., if \( \text{Vars}(E) = \emptyset \); \( \mathcal{P} \) is the set of all the agents, ranged over by \( P, Q, \ldots \).

The operational semantics of this calculus is defined by the transition relations \( \xrightarrow{\Delta} \), included in \( \mathcal{P} \times \mathcal{P} \), defined for each \( \delta \in \mathcal{A} \). This definition follows the structure of agents and is obtained with an inference system. The complete set of axioms and inference (or transition) rules (or more precisely inference schema) associated with the defined operators is given in Table 1 where \( \text{fix} X = E \) abbreviates the \( X \)-indexed family \( \langle \text{fix}_X X = E \rangle_{x \in X} \); if \( P \) is a \( Y \)-indexed family of agents and \( X \) is a set of variables with \( X \subseteq Y \), \( E[P/X] \) is obtained by the substitution in \( E \) of each free occurrence of the variable \( X \) by \( P_X \), for all \( X \in X \).

The semantics of an agent is given by the associated Labelled Transition System \( \text{LTS} = \langle \mathcal{P}, \mathcal{A} \mathcal{E} \mathcal{T}, T, \cdot \rangle \) where the transition relations \( T \) are derived from Table 1.

The equality relation links the agents which are weakly bisimilar in any context (see, e.g., [12] for a summary of the results on strong bisimilarity (denoted \( \sim \)), weak bisimilarity (denoted \( \approx \)) and equality (also called bisimulation congruence and denoted \( = \) or \( \approx \))

214
3. Derivation of enabling and disabling

Let us consider the followings two operators:

\[ E_1;_s E_2 \text{ (enabling)} \quad | \quad E_1[\_s E_2 \text{ (disabling)} \]

whose semantics are given by the inference rules in Table 2.

If one considers the enabling and disabling in LOTOS, it can be seen that our definition for these operators is more general. Indeed, in LOTOS, there is an action-denotation exit, which enables an action \( \delta \); this action indicates the termination of an agent; after a \( \delta \), the agent can no longer perform any action. In other words, in LOTUS, \( \delta \) actions have the strong termination property [1,7] for any agent \( P \): for any derivative \( P' \) of \( P \), if \( P' \Downarrow P'' \), then \( P'' \) cannot perform any action.

Since a \( \delta \) action may have parameters (offers), it will be mapped in our calculus on a set of actions \( D \) (such a mapping between LOTOS and a basic calculus is given, e.g., in [1]). Then, it may be seen that the LOTOS enabling (\( \langle \_ \rangle \)) and disabling (\( \langle \_ \rangle \)) operators correspond respectively to our \( ;_D \) and \([_D \rangle \) operators.

Since we want to use our calculus for theoretical investigations concerning LOTOS specifications, let us restrict our calculus to \( ;_D \) and \([_D \rangle \) operators where any \( d \in D \) has the strong termination property for all the considered agents.

### Table 1
Axioms and inference rules

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Summation</th>
<th>Composition</th>
<th>Hiding</th>
<th>Relabelling</th>
<th>Recursion</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a.P \Downarrow P) (in particular: (\tau.P \Downarrow P))</td>
<td>(\sum P \Downarrow P') ((P_j \in P))</td>
<td>(P_1 \Downarrow P_1') (P_1</td>
<td>_{_s} P_2 \Downarrow P_1</td>
<td>_{_s} P_2) (a \in S)</td>
<td>(P \Downarrow P') (P \Downarrow \Downarrow \Downarrow) (P \Downarrow P') (a \in S)</td>
</tr>
</tbody>
</table>

### Table 2
Axioms and inference rules

<table>
<thead>
<tr>
<th>Enabling</th>
<th>Disabling</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P_1 \Downarrow P_1') (P_1</td>
<td>_{_s} P_2 \Downarrow P_1</td>
</tr>
<tr>
<td>(P_1 \Downarrow P_1') (P_1</td>
<td>_{_s} P_2 \Downarrow P_1</td>
</tr>
</tbody>
</table>
We show below that the LOTOS enabling and disabling are derived operators in our calculus; let us recall that the desabling operator is generally considered as nonderivable in such a calculus with simple actions and this kind of operators [1].

The following propositions use the minimal sorting \(\alpha(P)\) of some agent \(P\): \(\alpha(P)\) gives an upper approximation, based on the syntax of the agent \(P\), of its alphabet (minimal sort) i.e., the set of actions that this agent and its derivatives may have (see [12] or [7] for a precise definition of \(\alpha(P)\)).

**Proposition [7].** If \(D \subseteq \mathcal{L}\), \(P_1, P_2 \in \mathcal{P}\) and

1. \(\forall d \in D\), \(d\) has the strong termination property for \(P_1\),
2. \(\Phi^1\) and \(\Phi^2\) are two relabelling functions such that:

\[
\forall d \in D: \Phi^2(\Phi^1(d)) = d, \quad \forall a \in (\mathcal{L} - D): \Phi^1(a) = a,
\]

then

\[
P_1 \cdot_D P_2 \sim (P_1 \cdot_D \sum_d \langle d \cdot (P_2[\Phi^1]) \rangle_{d \in D}) \setminus D[\Phi^2]
\]

and therefore both agents are equals, i.e., the second one can replace the first one in any context.

First, we may notice that since we have supposed that the agents studied here are the result of a mapping of some LOTOS specification, all these conditions can always be fulfilled. Indeed, we have supposed that condition (1) is always fulfilled. Moreover, we can always select a mapping between LOTOS and our calculus which leaves unused an infinite number of actions; then we can always find relabelling functions \(\Phi^1, \Phi^2\) satisfying (2).

To show that this proposition holds, let us denote

\[
(P_1 \cdot_D \sum_d \langle d \cdot (P_2[\Phi^1]) \rangle_{d \in D}) \setminus D[\Phi^2]
\]

by \(B[P_1, P_2]\) (\(B\) is a context with two distinct holes).

We can check that

- if \(P_1 \not\Rightarrow P_1'\) with \(a \in D\), then \(P_1 \cdot_D P_2 \not\Rightarrow P_1' \cdot_D P_2\) and \(B[P_1, P_2] \not\Rightarrow B[P_1', P_2]\); moreover, since \(\alpha(P_1') \subseteq \alpha(P_1)\), \(\Phi^1\) and \(\Phi^2\) still have the good properties w.r.t. \(P_1'\) and \(P_2\);

- if \(P_1 \not\Rightarrow P_1'\) with \(d \in D\), then \(P_1 \cdot_D P_2 \not\Rightarrow P_2\) and \(B[P_1, P_2] \not\Rightarrow B' = (P_1' \cdot_D P_2[\Phi^1]) \setminus D[\Phi^2]\) where, since \(d\) has the strong termination property for \(P_1\), \(P_1'\) cannot perform any action, i.e., \(P_1' \sim 0\); this implies from the definition of \(\Phi^1\) and \(\Phi^2\) that \(B' \sim P_2\).

On the other hand, these are the only possible evolutions of \(B[P_1, P_2]\) and of \(P_1 \cdot_D P_2\). \(\square\)

Now let us examine the disabling operator. As for the enabling, under some general conditions, the disabling operator may be derived from the existing ones:

**Proposition [7].** If \(D \subseteq \mathcal{L}\), \(P_1, P_2 \in \mathcal{P}\) and \(\Phi^1, \Phi^2\) are relabelling functions with \(\Phi^2(\alpha(P_2)) \cap (\alpha(P_1) \cup D) = \emptyset\)

and \(\forall a \in \alpha(P_2): \Phi^2(\alpha(a)) = a\) and \(\forall a \in \alpha(P_1): \Phi^2(a) = a\), then

\[
P_1 \cdot_D P_2 \equiv (P_1 \cdot_D \text{fix}_X = C) [\Phi^2],
\]

where \(X = \langle X_1, X_2 \rangle\) and \(C = \langle C_1, C_2 \rangle\) with

\[
C_1 = P_2[\Phi^1] + \sum \langle a . X_1 \rangle_{a \in \alpha(P_1) - D} + \sum \langle d . X_2 \rangle_{d \in D}, \quad C_2 = \sum \langle a . X_2 \rangle_{a \in \alpha(P_1)}.
\]
To show intuitively that the proposition is true, let us denote $\alpha(P_1) \cup D$ by $L$ and $(P_1 \mid_L \text{fix}_1 X = C)[\Phi^2]$ by $B[P_1, P_2]$.

Let us first notice that the delicate point is to find a relabelling $\Phi^1$ which is injective on $\alpha(P_2)$ and does not send it on any action in $\alpha(P_1) \cup D$: this is possible iff $|\alpha(P_2)| < |Q - (\alpha(P_1) \cup D)|$, which is always possible when defining a mapping between full LOTOS and our basic calculus.

Intuitively, the relabellings $\Phi^1$ and $\Phi^2$ allow to transform $\alpha(P_1)$ in such a form that it no longer has any intersection with $\alpha(P_2)$ and then to come back to the original configuration. To simplify our explanation, let us suppose that $\alpha(P_1) \cap \alpha(P_2) = \emptyset$; in this case we may take $\Phi^1$ and $\Phi^2$ as the identity relation and we may work with a slightly simplified version of $B[P_1, P_2]$, i.e., $B_\delta[P_1, P_2] = P_1 \mid_L \text{fix}_1 X = C$, where

$$C_1 = P_2 + \sum \langle a.X_1 \rangle_{a \in \alpha(P_1) - D} + \sum \langle d.X_2 \rangle_{d \in D}, \quad C_2 = \sum \langle a.X_2 \rangle_{a \in \alpha(P_1)}.$$ 

$B_\delta[P_1, P_2]$ is composed of two subagents, $P_1$ and $\text{fix}_1 X = C$ synchronized on $L = \alpha(P_1) \cup D$ so that $P_1$ may not evolve alone. If we look at the definition of $\text{fix}_1 X = C$, we can see that it plays the role of a controller/selector:

- if $P_1 \overset{a}{\rightarrow} P'_1$ with $a \in \alpha(P_1) - D$, then either $a = \tau$ or $a \neq \tau$ and $\text{fix}_1 X = C \rightarrow \text{fix}_1 X = C$, so that in either cases $B_\delta[P_1, P_2] \overset{a}{\rightarrow} P'_1 \mid_L \text{fix}_1 X = C$ and the controller/selector restarts. This corresponds to

$$\frac{P_1 \overset{a}{\rightarrow} P'_1}{P_1 \mid_D P_2 \overset{a}{\rightarrow} P'_1 \mid_D P_2} \quad (a \in D);$$

- if $P_1 \overset{d}{\rightarrow} P'_1$ with $d \in D$, then $\text{fix}_1 X = C \overset{d}{\rightarrow} \text{fix}_2 X = C$ and $B_\delta[P_1, P_2] \overset{d}{\rightarrow} P'_1 \mid_L \text{fix}_2 X = C$ which allows all actions of $P'_1$, so that the controller/selector “forgets” $P_2$ and releases its control on $P_1$ ($P'_1$). This corresponds to

$$\frac{P_1 \overset{d}{\rightarrow} P'_1}{P_1 \mid_D P_2 \overset{d}{\rightarrow} P'_1} \quad (d \in D);$$

- if $P_2 \overset{c}{\rightarrow} P'_2$, then $\text{fix}_1 X = C \overset{c}{\rightarrow} P'_2$ and $B_\delta[P_1, P_2] \overset{c}{\rightarrow} P'_1 \mid_L P'_2$ and since $\alpha(P'_2) \subseteq \alpha(P_2)$ has no intersection with $\alpha(P_1)$, $P_1$ can no longer perform any visible action (the disruption has occurred) and we have selected the evolutions of $P_2$. This corresponds to

$$\frac{P_2 \overset{c}{\rightarrow} P'_2}{P_1 \mid_D P_2 \overset{c}{\rightarrow} P'_2}.$$ 

Finally, we can see that no other evolution than the three cases described above can be performed by $B_\delta[P_1, P_2]$. $\square$

Note that, in this case, only bisimulation congruence (and not strong bisimulation) is proven between an agent in disabling form and the agent derived by our basic operators. This is due to the fact that our controller/selector cannot prevent $P_1$ to have internal actions after an action $P_2 \overset{c}{\rightarrow} P'_2$ leading to $P_1 \mid_D P'_2$.

We have proved that each time we meet an agent $P \equiv P_1 \mid_D P_2$ or $P \equiv P_1 \mid_D P_2$ ($P_1, P_2 \in \mathcal{P}$) arising from the translation of a LOTOS specification, we can find a basic context for $P_1, P_2$ replacing the enabling or disabling operator. Since this result uses the sort or sorting of $P_1$ and $P_2$, it cannot directly be applied to nonagent expressions $E_1 \mid_D E_2$ and $E_1 \mid_D E_2$ (with free variables). However, the result may be extended to agents $P$ containing enabling or/and disabling expressions, i.e., for some $P \equiv C[E_1 \mid_D E_2]$ or $P \equiv C[E_1 \mid_D E_2]$ [8]. Indeed, in that case, the free variables in $E_1 \mid_D E_2$ or $E_1 \mid_D E_2$ are bound by
known fix agents and therefore we can calculate the sorting of these expressions after the substitutions of their free variables by the corresponding fix agents.

4. Conclusion

We have defined a basic calculus for LOTOS which uses simple actions and only six kinds of basic operators.

If we compare this basic calculus with the other basic calculi on which LOTOS can be mapped (Basic LOTOS [2], LOTCAL [1], SCCS [11]), one of the advantages of our basic calculus is its simplicity. Firstly, it contains less operators than Basic LOTOS; in particular, the enabling and disabling operator have been removed. Secondly, our operators have simpler inference rules; in particular, we do not consider a special (set of) action(s) δ (exit). On the other hand, contrary to LOTCAL and SCCS, our basic calculus uses simple actions and thus seems closer to LOTOS.

In summary, the main point of this paper has been to show that, contrary to what was supposed in other works, it is possible to define a basic calculus with simple actions (contrary to LOTCAL [1] or SCCS [11]) where the disabling operator is derived from the basic ones.

Acknowledgment

The author would like to thank Professor DeviIlers, the director of its thesis, for reading and providing useful comments on a draft of this paper.

References