Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE
DELLA INFORMAZIONE

PISA

Principles for a Temporal Semantics of LOTOS

Alessandro Fantechi, Stefania Gnesi, Cosimo Laneve

Nota Interna B4-~~36~~ 37
Agosto 1988

PRINCIPLES FOR A TEMPORAL SEMANTICS OF LOTOS

A. Fantechi* , S. Gnesi*, C. Laneve**

* *Istituto di Elaborazione dell'Informazione - C.N.R. - Pisa - Italy*

** *Dipartimento di Matematica - Università di Siena - Italy*

Abstract. A temporal semantics for a subset of LOTOS, a language developed for the formal description of communication protocols, is presented. The semantics is given using a compositional approach by which it is possible to associate temporal logic formulas to each language construct. The considered subset covers the main characteristics of LOTOS for what concerns the concurrent aspects; the extension to full LOTOS is discussed.

1. Introduction

The temporal semantics of a language associates a temporal logic formula to each language construct and therefore gives the meaning of a program as a temporal logic formula. This can be used to verify that programs written in a concurrent programming language satisfy properties defined by temporal logic formulas.

The same principle can be applied to a language for the specification of concurrent systems; this paper studies in particular the application to LOTOS [DP 8807], a language based on process algebraic methods and derived from Milner's Calculus of Communicating Systems [Mil 80]. This will allow us to define a proof system for the verification that LOTOS specifications satisfy properties expressed by temporal logic formulas.

However, LOTOS has been given an operational semantics, and a formal notion of (observational) equivalence between LOTOS processes has been defined precisely. It is necessary therefore to show the relationships (possibly the consistency) of the given temporal semantics and the "official" LOTOS operational semantics.

As a first step in the direction of giving a temporal semantics for LOTOS, we have defined a subset of this language, which contains most of the interesting features; moreover, we will start following the easier approach to the problem by adopting the compositional approach method presented in [Bar 84, Bar 85, Bar 86], which use a classical linear time temporal logic and which allows a definition of the semantics in a denotational style.

This paper starts by an overview of the linear temporal logic used (Section 2); in Section 3 we present the subset of LOTOS which we will give a temporal semantics and which we consider enough powerful to capture all the main characteristics of this language. The compositional approach to temporal semantics for the subset of LOTOS processes is presented in Section 4. Section 5 discusses the relationships between the temporal semantics and the operational semantics of the language (the simple abstractness of the given temporal semantics with respect to the operational semantics modulo string equivalence is proved in [Fan 88]). Section 6 briefly discusses the introduction of justice

requirements in the proposed temporal semantics. Finally, the possibility of extending our semantics to full LOTOS is discussed (Section 7).

2. Temporal logic

A concurrent program can be identified by a set of execution traces (behaviours) σ which can be represented as:

$$\sigma : \; s_0 - {}^{a}1\rightarrow \; s_1 - {}^{a}2\rightarrow s_2 - {}^{a}3\rightarrow \cdots \qquad (1)$$

in which s_i are states of the program and a_i are transitions, i.e. executions of actions, which modify the state. In the interleaving model for concurrent programs it is assumed that a transition is exactly one atomic action of the program. This model, which is isomorphic to the natural numbers (i.e. discrete, ordered linearly and with a definite starting point), is used because linear execution traces and discrete time models are natural (operational) models of program execution behaviour.

Temporal logic is that branch of modal logic dealing with the evolution of situations over time, and therefore also on concurrent programs seen as transition systems.

In particular we will use a version of temporal logic which is capable of expressing properties of the sequence models seen above. This is a linear time temporal logic in which predicates specify events, i.e.transitions, and formulae contain universal and existential quantifiers (first order temporal logic). A transition predicate is true in the state where the event it represents happens. We will denote by the notation $p|_s$ the truth of predicate p in state s and, if σ is like defined in (1), with the notation σ^i the model :

$$\sigma^i : \; s_{i-1} - {}^{a}i \rightarrow \; s_i - {}^{a}i+1 \rightarrow s_{i+1} - {}^{a}i+2 \rightarrow \cdots$$

The operators that we will use are the usual first order logic connectives and quantifiers:

$$\textbf{true}, \; \textbf{false}, \; \neg, \; \vee, \; \wedge, \; \Rightarrow, \; \Leftrightarrow, \; \forall, \; \exists$$

together with temporal operators:

$$\circ \text{ (next)}, \; [] \text{ (always)}, \; \diamond \text{ (eventually)}, \; \mathcal{W} \text{ (unless)}, \; \mathcal{U} \text{ (until)}, \; [\,/\,] \text{ (relabelling)}$$

and a fixed point constructor:

$$\nu\xi.\, \chi\,(\xi)$$

denoting the *maximal* solution to $\xi \Rightarrow \chi\,(\xi)$. This last schema indicates that ξ is bound to the ν quantifier and the formula χ contains the free temporal variable ξ.

Using the model σ defined in (1), the interpretation of a formula ϕ is a function on the set of temporal formulae which yields a value in {t,f}. We will define this interpretation by induction on the

structure of ϕ:

$$\sigma \models \textbf{true} = \textbf{t}, \qquad \sigma \models \textbf{false} = \textbf{f}.$$

If a is a transition predicate, likewise, we define:

$$\sigma \models a = \begin{cases} \textbf{t} & \text{if } a_1 = a \\ \textbf{f} & \text{if } a_1 \neq a \end{cases}$$

As far as non atomic formulae are concerned their interpretation, assuming that in the metalanguage not, or, and, implies, exists, for all are defined, is:

$\sigma \models \neg\phi$	iff	not $(\sigma \models \phi)$
$\sigma \models \phi \vee \psi$	iff	$(\sigma \models \phi)$ or $(\sigma \models \psi)$
$\sigma \models \phi \wedge \psi$	iff	$(\sigma \models \phi)$ and $(\sigma \models \psi)$
$\sigma \models \phi \Rightarrow \psi$	iff	$\sigma \models \neg\phi \vee \psi$
$\sigma \models \phi \Leftrightarrow \psi$	iff	$\sigma \models (\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$
$\sigma \models \exists x.\phi(x)$	iff	exists t : $(\sigma \models \phi(t))$
$\sigma \models \forall x.\phi(x)$	iff	for all t : $(\sigma \models \phi(t))$

The variables bounded in the quantified formula range on a domain (different from that of the same formulae): for ours aims we may assume that this domain is the set *Nat*. Informally, we may summarize the above interpretations saying that the evaluation of boolean connectives is obtained by applying the same connectives to the evaluated subformulae. Temporal connectives deserve more attention:

$\sigma \models \circ\phi$	iff	$\sigma^i \models \phi \qquad i=2$

A formula ϕ in the scope of the *next* operator evaluates **true** in a state iff ϕ is **true** in the following state.

$\sigma \models []\phi$	iff	for all i : $(\sigma^i \models \phi)$

The unary operator *always* guarantees the truth of a formula in every state of the computation. In section 2 we have pointed out the need to specify *safety* properties for concurrent programs. These properties, which say that *something bad never happens,* can well be expressed using the *always* operator.

$\sigma \models \Diamond\phi$	iff	exists i : $(\sigma^i \models \phi)$

The unary operator *eventually* guarantees that a formula (something good) will become true. In this way it is possible to express liveness properties.

$$\sigma \models \phi \,\mathcal{U}\, \psi \qquad \text{iff} \qquad \underline{\text{exists}}\ k : (\sigma^k \models \psi)\ \underline{\text{and}}\ \underline{\text{for all}}\ i : 0 \le i < k\ \underline{\text{and}}\ (\sigma^i \models \phi)$$

Informally $\phi \,\mathcal{U}\, \psi$ is true in a state if ψ will eventually hold in the future and in every moment up to then ϕ will hold (it is not necessary that ϕ is true when ψ holds).

$$\sigma \models \phi \,\mathcal{W}\, \psi \qquad \text{iff} \qquad \sigma \models []\phi\ \ \underline{\text{or}}\ \ \sigma \models \phi \,\mathcal{U}\, \psi$$

This operator, also called *weak until*, is different from the preceding one as it does not require the eventual holding of the ψ formula.

$$\sigma \models \phi\,[a/b] \qquad \text{iff} \qquad \sigma' \models \phi\ \ \underline{\text{and}}\ \ \underline{\text{for all}}\ k>0\ \ \sigma^k \models a\ \ \underline{\text{implies}}\ \ \sigma'^k \models a$$

$\phi[a/b]$, where a and b are transition predicates, denotes the logical operator which has the effect of substituting every occurence of b in the model sequences of a formula ϕ with a. This operator, which is different from a syntactic substitution of every occurrence of b with a in the formula ϕ, has been named "relabelling"[Den 88] since it is analogous to the CCS operator with the same name.

$$\sigma \models \nu\xi.\, \chi(\xi) \qquad \text{iff} \qquad \underline{\text{for all}}\ k>0\ \ \sigma \models \chi^k(\mathbf{true})$$

where $\chi^k(\mathbf{true})$ is the temporal formula $\chi(\chi(\ldots\chi(\mathbf{true})\ldots))$ k-times. The fixed point constructor $\nu\xi.\,\chi(\xi)$ denotes the maximal solution to $\xi \Rightarrow \chi(\xi)$, which exists if and only if the function $\chi(\xi)$ is monotonic and continuous. The monotonicity requirement can be ensured by the appearance of the temporal formula ξ in χ under an even number of negations [Ban 86].

Interleaving constraint

We will use the presented temporal logic to give the semantics to a language which admits only interleaved executions, that is, in which actions are performed one at a time. Therefore we will deal only with model sequences in which only one transition predicate is true at any time.

For example the formula $((a \vee b) \wedge \circ[]c)$ is also satisfied by the model sequence which performs 'a' and 'b' concurrently as its first action. In order to adhere to the interleaving model, we must rule out this possibility. This can be achieved by requiring that every model considered satisfies the following predicate:

$$[] \bigwedge_{\substack{x,y \in \text{Action} \\ x \neq y}} \neg(x \wedge y)$$

where Action = $\{f\langle v\rangle \,/\, f \in \text{Gates} \wedge v \in \mathit{Nat}.\} \cup \{e,i\}$.

We will implicitly assume in the following that every formula is conjucted with the interleaving predicate.

Note that relaxing the interleaving constraint does not immediately produce a concurrent (e.g. "multiset") semantics; this would require a more careful defintion of some of the logic operators.

3. The Mini-LOTOS language

The subset of LOTOS, called Mini-LOTOS, that we consider in this paper mantains the main concurrent features of the original language; in particular, those features are maintained which still make this subset different from Milner's CCS: the form of interaction and the definition of parallel composition.

More precisely, the following restrictions with respect to LOTOS are operated:

1. Since our focus is on concurrency aspects, we ignore features related to the definition of abstract data types by means of algebraic specifications. In particular our attention is restricted to types which are only natural numbers (*Nat*, in the following).
2. Moreover, we do not consider the static structure of LOTOS programs; Mini-LOTOS programs have the following syntax:

   Program := *process declaration list ; behaviour expression*
   process declaration list := *process declaration ; process declaration list*
   process declaration := **process** *process identifier[gate*][value*]*
   ':=' behaviour expression **endproc**

   where *gate** and *value** are, respectively, a possibly empty list of gates and values (of type *Nat*).
3. The main semantic restriction is that no functionality is assigned to Mini-LOTOS processes, and exit and enabling operators are not considered: Mini-LOTOS processes terminate without returning any value. See Section 7 for a further discussion on this restriction.
4. We have not considered the disabling operator.
5. A minor semantic restriction is that we only allow a simple form of interaction: interactions in which both multiple value offer and simple value offer are present are not considered in order to simplify the definition of the synchronization rules.

Moreover we only allow one form of parallelism (that syntactically expressed by the operator |[*set of gates*]|). This parallelism is in any case the most general and includes the other two (syntactically expressed through || and |||) as degenerate cases (||≡|[*all gates*]| and |||≡|[∅]|).

In Table 1 we present the actions, operators and related their operational semantics of the Mini-LOTOS language. From the operational semantics we can observe that transitions of LOTOS programs are the observable actions (marked as o) and the unobservable i actions, and they only move the control point along the behaviour expression, without any other state change. We have not considered the "d" actions of LOTOS, due to the absence of exit and enabling operators.

operator	syntax	operational semantics	informal meaning
Inaction	**stop**		denotes a process which cannot perform any action.
Unobservable action	i;B	i;B -i->B	models an event internal to the process.
Observable action with multiple value offer:	g ?x;B	g ?x;B(x)-g<v>->B(v) *for any value v of sort Nat*	models a process which can perform any transition g<v> with v∈Nat.
with single value offer:	g !E;B	g !E;B-g<*val*(E)>->B	denotes a process which can transit only with transition g<*val*(E)>.The *val* operator simply evaluates its argument.
Choice	B1[]B2	B1-oi->B1' *implies* B1[]B2 -oi->B1' B2-oi->B2' *implies* B1[]B2 -oi->B2'	the actions of the process are the set of possible actions of B1 and B2.
Boolean guarding	[E] → B	B-oi->B' *and* val(E)=t *implies* [E] → B -oi-> B'	the process behaves like B if val(E)=t.
Parallel composition	B1|S|B2	B1-oi->B1' *and* gates(oi)∉S *implies* B1|S|B2-oi-> B1'|S|B2 B2-oi->B2' *and* gates(oi)∉S *implies* B1|S|B2-oi-> B1|S|B2' B1-o->B1' *and* B2-o->B2' *and* gates(o)∈S*implies* B1|S|B2-o-> B1'|S|B2'	the parallel composition forces the subprocesses to interact at every gate in the set S.
Hiding	**hide** S **in** B	B-oi->B' *and* gates(oi)∉S *implies* **hide** S **in** B-oi-> **hide** S **in** B' B-o->B' *and* gate(o) ∈S *implies* **hide** S **in** B-i-> **hide** S **in** B'	this operator allows the transitions at gates in S to be internalized.
Process instantiation	$p[g_1,...g_n][E_1,...E_m]$	$Bp[val(E_1),...val(E_m)][g_1/h_1,...g_n/h_n]$ -oi->B' *implies* $p[g_1,...g_n][E_1,...E_m]$ -oi->B'	the transitions of a process instantiation are those of the body of the process declaration (Bp) substituting formal parameters (h_i) with actual ones.

Tab.1

Note that this semantics does not distinguish between a process which is terminated and one which is deadlocked: i.e. their observational behaviour is the same.

4. The compositional approach

An approach, which can be used to associate to every process of a concurrent system a logical formula which axiomatically defines it, has been developed by Barringer, Kuiper and Pnueli [Bar 84, Bar 85]. The formula is very close to a behavioural specification of the system. In fact, these formulae

are given on the observable(1) sequence of actions performed by the system, using propositions which are considered true if the system is able to perform the corresponding action, and composing them with temporal operators to obtain formulae that can have an obvious interpretation in models which are sequences of actions. The idea is to characterize all behaviours prescribed by a program through one temporal logic formula. Consequently *a behaviour will be a program execution if and only if it satisfies the formula.* For example, the simple formula consisting of the proposition 'a' will be true for the processes which can perform immediately an 'a' action, while the formula $(a \vee b) \wedge \circ[]c$ will be true for the processes which can perform an action 'a' followed by an infinite sequence of 'c' actions or an action 'b' followed by an infinite sequence of 'c' actions. Recall that the interleaving constraint prevents this formula to have as a model a sequence which can perform concurrently 'a' and 'b'.

In order to obviate the problem of compositionality the semantics is given *open*, i.e. it describes the process immersed in all possible (parallel) environments. For this purpose, a special proposition '*e*', denoting an unknown environment action, is introduced (in [Bar 86] the same proposition is named 'i'; we choose 'e' to avoid confusion with the LOTOS 'i' (internal) action).

More precisely, the semantics is always given for closed systems consisting of the process in question and all possible environments. The semantics of a process in such closed systems is seen as a sequence of its own actions possibly interleaved with environment actions. When a process is composed with another one, its semantics is partially closed because part of the external environment of the first process becomes known, i.e. some of the environment actions of the first process will be the actions performed by the other.

Due to the compositionality of the approach, a temporal semantics can be given for a concurrent language by associating a temporal formula to each construct of the language with a set of syntax-directed clauses as it is usual when giving denotational semantics. In [Bar 86] the compositional approach has been used to give the semantics of several example concurrent languages; in the following section we will show how this method can be applied to Mini-LOTOS.

5. A compositional temporal semantics for Mini-LOTOS

The kernel of the temporal semantics is the function $\mathcal{M}$ which associates a meaning to every language construct, in a denotational style.

The language constructs are in LOTOS behaviour expressions, and their meaning is taken to be a predicate on possible execution sequences spawning from that behaviour expression. The meaning is however dependent on the context in which the construct is inserted, i.e. an environment which associates a process denotation to every declared identifier is considered as a parameter of the function $\mathcal{M}$. We adopt the notation $\mathcal{M}_\theta$ to indicate the environment parameter.

---

(1) In this context we intend by observable all that can be seen of the behaviour of the system at the chosen abstraction level. In particular, in the following, this notion of observable will include the "unobservable" actions 'i' of LOTOS, since we will be at a level of abstraction in which we can also observe the internal actions of a system.

The type of the semantic function is hence:

$$\mathcal{M} : S \rightarrow E \rightarrow TL$$

where S is the space of Mini-LOTOS behaviour expressionss, TL is the space of temporal formulae and E is the space of environments, i.e. of mappings from process identifiers to temporal logic formulae: $E : P \rightarrow TL$. Notice that in the following the word "environment" is overloaded with two meanings: the first denotes an element of E, the second distinguishes the "e" action as an environment action, in the typical sense of the compositional approach. The meaning is, however, obvious from the context each time.

The following are the clauses defining by structural induction the function $\mathcal{M}$, each followed by a comment.

$$\mathcal{M}_\theta(\textbf{stop}) = e\ \mathcal{W}\ \neg\circ\ \textbf{true}$$

The meaning of an inaction behaviour is true for any (possibly infinite) sequence of environment actions. '$\neg\circ$ **true**' is valid for any void sequence. The use of $\mathcal{W}$ operator in this clause is due to the fact that the process '**stop**' will execute transitions only if the environment is able to perform them: the process will terminate when also the environment will terminate

$$\mathcal{M}_\theta(i;B) = e\ \mathcal{W}\ [\, i \wedge \circ \mathcal{M}_\theta(B)\,]$$

The meaning of an unobservable action followed by a behaviour B is true for every finite sequence of environment events followed by the unobservable action and by sequences for which the meaning of B is true, or for the infinite sequence of e's..

$$\mathcal{M}_\theta(g!v;\ B) = e\ \mathcal{W}\ [\, g\langle v\rangle \wedge \circ \mathcal{M}_\theta(B)\,]$$

$$\mathcal{M}_\theta(g?x;\ B(x)) = e\ \mathcal{W}\ [\, \exists\, a \in Nat : g\langle a\rangle \wedge \circ \mathcal{M}_\theta(B(a))\,]$$

The meaning of an observable action followed by a behaviour B is true for every sequence of environment events followed by the observable action and by sequences for which the meaning of B is true. In this case it is possible that an interaction never occurs, since its occurrence is dependent by the environment avilability to perform the same action. Note that after an input interaction the behaviour can be dependent on the value exchanged in the interaction.

$$\mathcal{M}_\theta([b]\rightarrow B) = \begin{cases} b \rightarrow & \mathcal{M}_\theta(B) \\ \neg b \rightarrow & e\ \mathcal{W}\ \neg\circ\ \textbf{true} \end{cases}$$

The boolean guarded behaviour has the same meaning as B if the boolean condition is true, as **stop** if it is false.

$$\mathcal{M}_\theta(B_1 \square B_2) = \mathcal{M}_\theta(B_1) \vee \mathcal{M}_\theta(B_2)$$

The meaning of the choice operator is true if the meaning of one of the component behaviours is true.

In the following formula we should have used two functions: $\underline{1}$ and $\underline{2}$ defined on the set of gates Γ and wich give values, respectively, in the sets $\Gamma\times\{1\}$ and $\Gamma\times\{2\}$, such that $\underline{1}(x) = \langle x,1\rangle$, and $\underline{2}(x) = \langle x,2\rangle$. However, for notation simplicity, we will use subscripts to the names of the gates (that is, x_1, x_2) to mean the application of these functions.

$$\mathcal{M}_\theta(B_1 |g| B_2) =$$
$$\Big(\mathcal{M}_\theta(B_1)\,[(e \vee i_2 \vee \bigvee_{f\notin g, f\langle v\rangle \in \alpha(B_2)} f_2\langle v\rangle\ ) / e,\ i_1 / i,\ \forall f\notin g, f\langle v\rangle\in\alpha(B_1)\colon f_1\langle v\rangle / f\langle v\rangle] \wedge$$
$$\wedge\, \mathcal{M}_\theta(B_2)\,[(e \vee i_1 \vee \bigvee_{f\notin g, f\langle v\rangle \in \alpha(B_1)} f_1\langle v\rangle\ ) / e, i_2 / i,\ \forall f\notin g, f\langle v\rangle\in\alpha(B_2)\colon f_2\langle v\rangle / f\langle v\rangle] \Big)$$
$$[\forall f\notin g\ \ f\langle v\rangle / f_1\langle v\rangle,\ \forall f\notin g\ \ f\langle v\rangle / f_2\langle v\rangle,\ i / i_1, i / i_2\ ]$$

The logic relabelling operator is here liberally extended to express a set of substitutions. The last substitution applies to the result of the conjunction. Function α, defined on behaviour expressions, gives the set of transitions that a behaviour expression *may* perform during its execution (this set of transitions can be produced by a static analysis of the behaviour expression - actually, the static analysis produces a larger set of that strictly needed, but due to the use of subscripts, all unnecessary f<v>, that is those which are not present in the matching sequence, will be dropped by the conjunction operation).

Parallel composition partially closes the open semantics of the component processes. In order to explain the formula we should consider carefully the exact meaning of the LOTOS parallel composition: actions of the composed processes are interleaved, except for those actions occurring at gates belonging to the set 'g': these actions are performed simultaneously by both processes. Hence, given the various actions that the component processes can perform, we can distinguish the following cases:

1. If B_1 performs an *i* action, this is considered an environment action for B_2, thus an *i* in B_1 must be paired with an *e* in B_2 (and viceversa).
2. If B_1 performs a f<v> action, with $f\notin g$, this is again considered an environment action for B_2; thus it must be paired with an *e* in B_2 (and viceversa).
3. If B_1 performs a f<v> action, with $f\in g$, B_2 should also perform this action, and thus the same actions must be paired for B_1 and B_2.
4. Since the parallel composition should continue to have an open semantics, it must be open to environment actions which are considered as such for both B_1 and B_2: thus an *e* in B_1 should be paired with an *e* in B_2.

This pairing is performed by the conjunction of the component meanings. Suppose the next action performed by B_1 is 'x' and by B_2 is 'y'; this means that

$$\mathcal{M}(B_1) \Rightarrow x \qquad \text{and} \qquad \mathcal{M}(B_2) \Rightarrow y$$

hence

$$\mathcal{M}(B_1) \wedge \mathcal{M}(B_2) \Rightarrow x \wedge y \qquad (2)$$

Now, if x and y are different, formula (2) will have no model due to the interleaving constraints. Hence we should restrict models to the case when x and y are the same. This is satisfactory for *e* actions (case 4) and for actions on gates in 'g' (case 3). In the other cases, we should substitute one of the *e*'s in $\mathcal{M}(B_1)$ with an *i* (case 1) or with a *f*<*v*> (f∉ g) in case 2. This substitution amounts to partially closing the open semantics of B_1 with actions performed by B_2.

Summarizing what has been said so far till now would produce the following simpler formula which expresses the temporal meaning of $B_1|g|B_2$:

$$\mathcal{M}_\theta(B_1)[(e \vee i \vee \bigvee_{f \notin g, f<v> \in \alpha(B_2)} f<v>)/e] \wedge \mathcal{M}_\theta(B_2)[(e \vee i \vee \bigvee_{f \notin g, f<v> \in \alpha(B_1)} f<v>)/e]$$

However this formula can confuse actions performed by B_1 with the same actions performed by B_2 (in LOTOS we have no "ownership" on actions: any process can perform any action at any gate). To overcome this problem we must identify (with a substitution) the owner of the (i and f<v>, f∉ g) actions. The identity of the owner is then dropped from the formula describing the behaviour of the composed process.

$$\mathcal{M}_\theta(\textbf{hide g in B}) = \mathcal{M}_\theta(B)\ [\ \forall f \in g: i/f<v>]$$

The meaning of hiding is true for every sequence which satisfies the meaning of its operand, in which every occurrence of the actions is substituted by unobservable actions.

$$\mathcal{M}_\theta(p([a][v])) = \theta(p)\ [a/gates(p), v/values(p)]$$

The meaning of a process call is retrieved by the environment, with the obvious parameter substitutions - here the substitution symbol is used to denote the usual parameter substitution.

It is now time to define how an environment θ is built for a program. This is achieved by a semantic function $\mathfrak{D}$ which is defined on a list of process declarations. The type of $\mathfrak{D}$ is:

$$\mathfrak{D} : Decl^* \rightarrow E$$

and is defined in the following way (using an auxiliary function $\mathfrak{D}_1 : Decl^* \times E \rightarrow E$):

$$\mathfrak{D}(decls) = \nu\xi.\, \mathfrak{D}_1(decls)(\xi)$$
$$\mathfrak{D}_1(\textbf{process } p[g][v] := B \textbf{ endproc}; decls)(\xi) = \{<p[g][v], \mathcal{M}_\xi(B)>\} \cup \mathfrak{D}_1(decls)(\xi)$$
$$\mathfrak{D}_1(nil)(\xi) = \emptyset$$

The definition of $\mathfrak{D}_1$ augments the mapping with a new pair for every declaration. The pairs consist of a process heading and the meaning of the corresponding behaviour expression. Since process calls can be recursive, the meaning of behaviours should also be evaluated in the complete environment.

The existence of the maximum fixed point is guaranteed by the consideration that the temporal logic formulae present in ξ can appear only under an even number of negations.

Finally, the program semantics is given by a semantic function $\mathcal{P}$ whose type is:

$$\mathcal{P} : Prog \rightarrow TL$$

with the following definition:

$$\mathcal{P}\ (\ \text{decls}\ ;\ \text{B}) = \mathcal{M}_{\mathcal{D}(\text{decls})}(\text{B})$$

6. Relationships between temporal and operational semantics.

In order to relate the given semantics to the operational one - given for LOTOS in an official standard document [DP 8807] we need to study the relationship between the Mini-LOTOS temporal semantics given in this work and its operational semantics (see Tab.1), derived from that of LOTOS.

The operational semantics of a Mini-LOTOS process is a labelled transition system (constituted by a set of states, a set of labels - the actions - and a transition relation defined over them). This semantics induces an obvious equivalence relation between processes: two processes are equivalent if the relative transition systems are the same.

However, we may strenghten the above equivalence relation, considering equivalent - that is, having the same semantics - two processes whenever no external observer can tell any difference between their behaviours is captured by the definition of an equivalence relation between transition system (observational equivalence [Mil 80]). Actually, this is the equivalence relation defined for the "official" LOTOS operational semantics.

On the other hand, the temporal semantics of a Mini-LOTOS process is a temporal logic formula; the possible executions of the process are all the ordered sequences of actions which satisfy this formula.

The use of such semantics implies another equivalence relation: two processes have the same meaning if the associated formulas are logically equivalent. The equivalence between processes so obtained is that called "string equivalence" [Hoa 81].

The fundamental difference between the observational and the string equivalence is that the equivalence classes are respectively, classes of action labelled trees (the synchronization trees of [Mil 80]) and classes of sets of sequences of actions. This is due to the basic model for the process used by the operational semantics, that is, the labelled transition system, contrasted by the sequence model used for linear time temporal logic. This causes their different discriminating power: the observational equivalence has a finer discriminating power with respect to the string one.

To acquire again the discriminating power of the observational equivalence the temporal semantics need the use of a branching time temporal logic, in which models are trees of actions. This is a direction of future research for the specific case of LOTOS.

All the discussion above has been made more precise for the case of Mini-LOTOS in [Fan 88], in which the operational and the temporal semantics are formally compared. That work shows that the

given temporal semantics is simply abstract with respect to the operational semantics modulo string equivalence.

7. Justice requirements

The point in giving temporal semantics is not, anyway, in discussing discriminating power. The given semantics, being fully abstract with respect to maximal trace semantics, does not give any other advantage than being able, by working on logical formulae, to prove that a LOTOS program satisfies a property given as a temporal logic formula.

An interesting aspect to be discussed is instead whether, in giving a temporal semantics to LOTOS, we are able to say something more, which cannot be expressed by the operational semantics. Something more offered by temporal logic is the possibility to express non continuous properties, such as *fairness*. To impose a fairness requirement means to cancel from a set of infinite sequences which are the models of a program those sequences which are not fair, by a conjunction with a proper formula expressing fairness constraints. An example of formula expressing a global fairness constraint is the following:

$$\bigwedge_{S \in \mathcal{P}(\mathrm{Action})} \left(\Box\Diamond o \left(\bigvee_{a \in S} a \right) \Rightarrow \bigwedge_{a \in S} \Box\Diamond a \right)$$

which states that, if a choice among a set of actions occurs infinitely often, than all the actions in the set must be performed infinitely often (actually, the above formula works only for "pure" LOTOS, without values).

Another property that can be expressed in general for LOTOS programs is *justice,* that is the requirement that if a process is able to perform a set of actions, an action of the set will eventually be performed.

Possible justice requirements, coherent with the basic principles of action synchronization in LOTOS, are the following:

- if a process is able to perform an external action, since its occurrence will in general depend on the external environment, it may be the case that the action is never performed (suppose for example that the process is synchronised with a non-terminating process which does not perform the same action).
- if a process is able to perform an internal action, eventually it must perform it.

The second requirement is expressible by modifying the formula expressing the semantics of internal action:

$$\mathcal{M}_\theta(i;B) = e\, \mathcal{U}\, [\, i \wedge o\mathcal{M}_\theta(B)\,]$$

while the semantics of observable actions may remain unchanged due to the first requirement.

Anyway, the unobservable actions are generated also by the hiding operator. Whenever an action is hidden, its occurrence is no more dependent from the environment and must eventually happen, for

the second requirement above. In order to express this, the formula expressing the semantics of hiding should be modified so that *until* operators instead of *unless* ones appear before any occurrence of a hidden action.

Another comment: consider the process:

a!3; **stop** [] i; **stop**

its temporal meaning, following the given temporal semantics, is:

$$(e\ \mathcal{W}\ (a\langle 3\rangle \wedge \circ\ (e\ \mathcal{W}\ \neg\circ\ \mathbf{true}))) \vee (e\ \mathcal{U}\ (i \wedge \circ\ (e\ \mathcal{W}\ \neg\circ\ \mathbf{true}))) =$$
$$= e\ \mathcal{W}\ ((a\langle 3\rangle \vee i) \wedge \circ\ (e\ \mathcal{W}\ \neg\circ\ \mathbf{true}))$$

which means that either action a<3> or i may occurr, but they also may never occurr. To enforce the second justice requirement in this case requires that the meaning of nondeterministic choice is modified in order to "spread" justice to the possible observable actions. The desired meaning would therefore be:

$$e\ \mathcal{U}\ ((a\langle 3\rangle \vee i) \wedge \circ\ (e\ \mathcal{W}\ \neg\circ\ \mathbf{true}))$$

It is an open question how to express the new meanings of hiding and choice by temporal semantics clauses and whether modifications are needed to the semantics of other LOTOS constructs. It is also not clear if it is possible to write a compositional justice semantics at all.

8. Towards a temporal semantics for full LOTOS

The main semantic restriction imposed on Mini-LOTOS is that we have not assigned a functionality to processes and we have, consequently, not introduced the *exit* and *enabling* operators. In order to give a meaning to such constructs, we need to be able to describe the sequential composition of sequences of actions. This can be done quite easily by adding a new operator to our temporal logic called "chop" or "combine" [Bar 84] and defined on execution traces σ as follows:

$$\sigma \models \phi\ \mathbb{C}\ \psi \qquad \text{iff} \qquad \text{exists } i:\ \sigma \models \phi \ \text{ and } \ \sigma^i \models \psi$$

(Its models are the sequences of states that are composed by a prefix which satisfies ϕ and a suffix which satisfies ψ).

However problems are raised by the use of this operator because of the increased complexity of the proofs of formulae which use it in conjunction with the fixed point operator [Bar 84].

We have neither given semantics to the LOTOS *disabling* operator. Actually, it seems that also this operator needs to introduce a new logic operator, which we call "cut", defined on execution traces σ as follows:

$$\sigma \models \phi\ [\rangle\ \psi \qquad \text{iff} \qquad \sigma = \sigma^1 \cdot \sigma^2 \text{ and exists } \sigma^3 \text{ such that } \sigma^1 \cdot \sigma^3 \models \phi \text{ and } \sigma^2 \models \psi$$
$$\text{or } \sigma \models \psi$$

(Its models are the sequences of states in which a prefix is the same of a prefix of a sequence which satisfies ϕ and the suffix satisfies ψ).

The effect of inserting the new operators in our temporal logic is still to be studied.

The other restrictions seem to need no conceptual novelty; multiple interaction can only make the meaning of the parallel composition more cumbersome. We can note however that in order to describe compositionally the semantics of many language operators we need to introduce specific logic operators which "mimic" the behaviour of the language ones (another example in this sense is the need to introduce the relabelling logic operator).

We have observed that a simpler form could be given to express the meaning of parallel composition, if the language does not allow the same action to be performed by the processes. This is in fact the classical problem of communication channels shared by more than one process; the use of the shared channel introduces hidden forms of nondeterminism. This problem has sometimes been solved in concurrent languages by forbidding such a facility, partly because of the increased implementation difficulty.

9. Conclusions

By providing a subset of LOTOS with temporal semantics, we have been able to capture the main characteristics of this language. We have however pointed out several open questions, some due to the desire of expressing non continuous properties of infinite LOTOS processes, other due to the extension of our work to cover full LOTOS.

We want also recall another open question, which at a first sight seems quite interesting in order to fully understand the relationships between temporal and operational semantics: the temporal meaning of non "well guarded" LOTOS terms.

Furthermore the real application of the temporal semantics to the proof of properties of complex LOTOS specifications should be evaluated.

References

[Ban 86] B. Banieqbal, H. Barringer: "A Study of an Extended Temporal Language and a Temporal Fixed Point Calculus", University of Manchester, Technical Report UMCS-86-11-1, October 1986.

[Bar 84] H. Barringer, R. Kuiper, A.Pnueli: "Now you may Compose Temporal Logic Specifications", Proc. 16th ACM Symposium on the Theory of Computing, 1984.

[Bar 85] H. Barringer, R. Kuiper, A.Pnueli: "A Compositional Temporal Approach to a CSP-like Language", in E.J. Neuhold and G. Chroust eds., Formal Models of Programming, IFIP, North-Holland, pp. 207-227.

[Bar 86] H. Barringer, "Using Temporal Logic in the Compositional Specifications of Concurrent Systems", University of Manchester, Technical Report UMCS-86-10-1, October 1986.

[Den 88] R. De Nicola, D. Latella, "Basic Ideas for Temporal Logic Semantics of a subset of TCCS", Draft, June 1988.

[DP 8807] ISO , "LOTOS, a Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", Second Draft Proposed Standard 8807, 1986.

[Fan 88] A. Fantechi, S. Gnesi, C. Laneve, "A Proof of fully abstractness for the temporal semantics of LOTOS", IEI Internal Report B4-38, August 1988.

[Hoa 81] Hoare, C.A.R., "A Model for Communicating Sequential Processes" Technical Monograph Prg-22, Computing Laboratory, University of Oxford, 1981.

[Mil 80] R. Milner : "A Calculus of Communicating Systems", Lecture Notes in Computer Science vol. 92, 1980.