*Consiglio Nazionale delle Ricerche*

# ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

**PISA**

## Transformation Rules

*Chapter 15 of*
*"Catalogue of LOTOS Correctness Preserving Transformations"*
*ESPRIT Project 2304 LOTOSPHERE*
*task 1.2 Third Deliverable*

R. De Nicola, A. Fantechi, S. Gnesi, P.Inverardi, M. Nesi

Nota Interna B4-31
Luglio 1992

# Chapter 15

# Transformation Rules

## 15.1 Definitions

The following are some definitions of equivalence and preorder relations between (Basic) LOTOS processes, used in this appendix and elsewhere in the Catalogue. These relations are defined using the notion of Labelled Transition System.

A Labelled transition System (LTS) is a 4-uple $(S, Act, \{R_x, x \in Act\}, s_0)$, such that S is a set of states, Act is a set of actions, $R_x \subseteq S \times S$, $s_0 \in S$.

We will use the notation $P \xrightarrow{a} Q$ to mean that $(P, Q) \in R_a$ and we will say that the system in the state P is able to perform the action $a$ and transform in the state Q.

The operational semantics of LOTOS associates a LTS to each process by means of rules of inference [9]. The set of actions Act for Basic LOTOS can be defined as $Gates \cup \{i\} \cup \{d\}$. The action $i$, the unobservable action, is treated in a special way by the "weak" equivalences. For their definition, a different relation between states of a LTS is needed:

$P \stackrel{a}{\Rightarrow} Q$ if and only if $\exists S_1, S_2, ...S_n \in$ S, such that $P \xrightarrow{i} S_1 \xrightarrow{i} S_2, ...S_k \xrightarrow{a} S_{k+1} \xrightarrow{i} ...S_n$

The two transition relations can be extended to sequences of actions: given $\sigma = a_1 \cdot a_2 \cdot ... a_n$,

$$P \xrightarrow{\sigma} Q \text{ iff } P \xrightarrow{a_1} S_1 \xrightarrow{a_2} S_2 ... \xrightarrow{a_n} Q$$

$$P \stackrel{\sigma}{\Rightarrow} Q \text{ iff } P \stackrel{a_1}{\Rightarrow} S_1 \stackrel{a_2}{\Rightarrow} S_2 ... \stackrel{a_n}{\Rightarrow} Q$$

### Bisimulation Congruence

A *bisimulation* $\mathcal{R}$ relation is a binary relation on S such that whenever $P\mathcal{R}Q$ and $a \in Act$ then:

$$i)\ P \xrightarrow{a} P' \Rightarrow \exists Q'.\ Q \xrightarrow{a} Q' \text{ and } P'\mathcal{R}Q';$$

$$ii)\ Q \xrightarrow{a} Q' \Rightarrow \exists P'.\ P \xrightarrow{a} P' \ and\ Q'\mathcal{R}P';$$

Two processes B1 and B2 are said *bisimulation equivalent* (written B1∼B2) if and only if there exists a bisimulation relating the initial states of their LTSs.

The bisimulation equivalence is actually a congruence (also called *strong congruence*).

### Observational Congruence

A *weak bisimulation* $\mathcal{R}$ relation is a binary relation on S such that whenever $P\mathcal{R}Q$ and $a \in Act$ then:

$$i)\ P \stackrel{a}{\Rightarrow} P' \Rightarrow \exists Q'.\ Q \stackrel{a}{\Rightarrow} Q' \ and\ P'\mathcal{R}Q';$$

$$ii)\ Q \stackrel{a}{\Rightarrow} Q' \Rightarrow \exists P'.\ P \stackrel{a}{\Rightarrow} P' \ and\ Q'\mathcal{R}P';$$

Two processes B1 and B2 are said *observationally equivalent* (or *weakly bisimular*, written B1≈B2) if and only if there exists a bisimulation relating the initial states of their LTSs.

We call *observational congruence* the largest congruence included in the bisimulation equivalence.

### Trace Congruence

The set $\{\sigma \mid \sigma \in Act^*, s_0 \stackrel{\sigma}{\Rightarrow} s_1\}$, where $s_0$ is the initial state of the LTS associated to a process P, is called the set of traces of P, written $Tr(P)$.

Two processes are said to be *trace equivalent* if they have the same set of traces. This equivalence is also a congruence (*trace congruence*).

### Reduction Preorder

We define the reduction preorder between two processes B1 and B2 in the following way [10]:

B1 *red* B2 if:

$i)\ Tr(B1) \subseteq Tr(B2)$

$ii)\ \forall \sigma \in Tr(B1)\ and\ \forall A \subseteq Act - \{i\},$

$$if\ \exists B1',\ B1 \stackrel{\sigma}{\Rightarrow} B1'\ and\ \nexists C:\ B1' \stackrel{a}{\Rightarrow} C\ \forall a \in A,$$

$$then\ \exists B2',\ B2 \stackrel{\sigma}{\Rightarrow} B2'\ and\ \nexists C:\ B2' \stackrel{a}{\Rightarrow} C\ \forall a \in A,$$

### Extension Preorder

We define the extension preorder between two processes B1 and B2 in the following way [10]:

B1 *ext* B2 if:

*i)* $Tr(B1) \supseteq Tr(B2)$

*ii)* $\forall \sigma \in Tr(B2)$ *and* $\forall A \subseteq Act - \{i\}$,

$if \ \exists B1', \ B1 \stackrel{\sigma}{\Rightarrow} B1' \ and \ \nexists C : \ B1' \stackrel{a}{\Rightarrow} C \ \forall a \in A,$

$then \ \exists B2', \ B2 \stackrel{\sigma}{\Rightarrow} B2' \ and \ \nexists C : \ B2' \stackrel{a}{\Rightarrow} C \ \forall a \in A,$

## 15.2 Laws for Strong, Observational and Trace Congruence

In the following a complete set of laws for strong, observational and trace congruence on finite basic LOTOS (i.e., Basic LOTOS without recursive process instantiations) is listed. This means that all transformations preserving one of these equivalences on Basic LOTOS can be ultimately reduced to a combination of these laws.

We begin with the laws for strong congruence, grouped together for each operator. We then present additional axioms for observational and trace congruence, respectively.

The laws are valid for Full LOTOS, but do not form a complete set of laws for the relevant equivalence on it, since they do not take into account value-passing. The set of laws for strong congruence is complete for Basic LOTOS without recursive process instantiations, but is not minimal; for a minimal complete set and its completeness proof we refer to [8]. The laws belonging to this minimal set has been marked with a $\star$ in the following. The set of laws for strong congruence includes instead all the laws presented in [9].

The additional axioms maintain the completeness with respect to the relevant equivalence.

### Exit

$exit = \delta; stop$ $\star$

### Choice []

B1 [] B2 = B2 [] B1 $\star$

B1 [] (B2 [] B3) = (B1 [] B2) [] B3 $\star$

B [] stop = B ⋆

B [] B = B ⋆

### Parallel |

(any of the following laws is a shorthand for the three laws obtained by replacing every occurrence of the symbol | with one of the following operators: $|[g1, ..., gn]|, |||, ||$ )

$B1|B2 = B2|B1$

$B1|(B2|B3) = (B1|B2)|B3$

$B1|[A]|B2 = B1|[A']|B2$ if A' and A define the same set of gates

$B1|[A]|B2 = B1|[A']|B2$ $A' = A \cap (L(B1) \cup L(B2))$

$B1|[A]|B2 = B1||B2$ $if A \supseteq (L(B1) \cup L(B2))$ ⋆

$B1|[\,]|B2 = B1|||B2$ ⋆

### Enabling >>

$stop >> B = stop$ ⋆

$exit >> B = i; B$

$(B1 >> B2) >> B3 = B1 >> (B2 >> B3)$

$(B1[]B2) >> B3 = (B1 >> B3)[](B2 >> B3)$ ⋆

$(a; B1) >> B2 = a; (B1 >> B2)$ if a$\neq \delta$ ⋆

$(a; B1) >> B2 = i; B2$ if a$= \delta$ ⋆

$B >> stop = B|||stop$

### Disabling [>

$B1[> (B2[> B3) = (B1[> B2)[> B3$

$B[> stop = B$

$(B1[> B2)[]B2 = B1[> B2$

$stop[> B = B$ ⋆

$exit[> B = exit[]B$

$(B1[]B2)[> B3 = (B1[> B3)[](B2[> B3)$ ⋆

$(a; B1)[> B2 = B2[]a; (B1[> B2)$ if a$\neq \delta$ ⋆

$(a; B1)[> B2 = (a; B1)[]B2$ if a$= \delta$ ⋆

### Hiding hide in

hide A in stop = stop ⋆

hide A in (a; B) = a; (hide A in B) if a$\notin A$ ⋆

hide A in (a; B) = i; (hide A in B) if a$\in A$ ⋆

hide A in B = hide A' in B if A' and A define the same set of gates

hide A in B = hide A' in B if A' = $A \cap L(B)$

hide A in hide A' in B = hide A" in B if A" = $A \cup A'$

hide A in B = B if $A \cap L(B) = \emptyset$

hide A in B1 [] B2 = (hide A in B1) [] (hide A in B2) ⋆

hide A in (B1 >> B2) = (hide A in B1) >> (hide A in B2)

hide A in (B1 [> B2) = (hide A in B1) [> (hide A in B2)

(the following law is a shorthand for the three laws obtained by replacing every occurrence of the symbol | with one of the following operators: $|[g1, ..., gn]|, |||, ||$ )

hide A in (B1 | [A'] | B2) = (hide A in B1) | [A'] | (hide A in B2) if $A \cap A' = \emptyset$

### Relabelling [S]

Note that in LOTOS no explicit operator for relabelling is provided, the effect of relabelling being achieved by means of the gate parameter passing in process instantiation. Therefore, for the sake of simplicity, we give the axioms by using the following notation: $[S] = [a1/g_1, ..., a_n/g_n], S(g_i) = a_i, S(g) = g \; if \; g \neq g_i \; for \; i = 1, .., n$

$stop[S] = stop$ ⋆

$exit[S] = exit$

$(a; B)[S] = S(a); B[S]$ ⋆

$(B1[]B2)[S] = B1[S][]B2[S]$ ⋆

$(B1|[A]|B2)[S] = B1[S]|[S(A)]|B2[S]$ if S is injective on $L(B1) \cup L(B2) \cup A$

$(B1 >> B2)[S] = B1[S] >> B2[S]$

$(B1[> B2)[S] = B1[S][> B2[S]$

(hide A' in B) [S] = hide A in B[S] if S is injective on $L(B) \cup A'$ and S(A') = A

B[S] = B if S = identity on L(B)

B[S1] = B[S2] if S1(a) = S2(a) for every $a \in L(B)$

B[S1] [S2] = B[S2oS1]

**Expansion Theorems**

Notation: $B1[]B2[]....[]Bn = []\{B1, ..., Bn\} = []S$ where $S = \{B1, ..., Bn\}$

Hp. every element of S has the structure $b_i; B_i$ .

If $B = []\{b_i; B_i | i \in I\}$ and $C = []\{c_j; C_j | j \in J\}$, then

$$B|[A]|C = \quad []\{b_i; (B_i|[A]|C) \mid name(b_i) \notin A, i \in I\}$$
$$[] \; []\{c_j; (B|[A]|C_j) \mid name(c_j) \notin A, j \in J\}$$
$$[] \; []\{a; (B_i|[A]|C_j) \mid a = b_i = c_j, name(a) \in A, i \in I, j \in J\} \qquad \star$$

$$B[> C = \quad C \; [] \; []\{b_i; (B_i[> C) | i \in I\}$$

$$hide \; A \; in \; B = \quad []\{b_i; \; hide \; A \; in \; B_i \mid name(b_i) \notin A, i \in I\}$$
$$[] \; []\{i; \; hide \; A \; in \; B_i \mid name(b_i) \in A, i \in I\}$$

$$B[S] = []\{S(b_i); B_i[S] | i \in I\}$$

### 15.2.1 Observational Congruence (i-laws)

By adding the following laws to the previous ones, a complete set of axioms for the observational congruence for finite LOTOS is obtained.

a ; i ; B = a ; B

B [] i ; B = i ; B

a ; ( B1 [] i ; B2) [] a ; B2 = a ; ( B1 [] i ; B2)

### 15.2.2 Trace Congruence Laws

By adding instead the following laws, a complete set of axioms for the trace congruence for finite LOTOS is obtained (note that the observational congruence laws for the internal action i can be easily derived by the following laws).

i;B = B

a;(B1 [] B2) = a;B1 [] a;B2

The following is a useful derived law for trace congruence:

B || B = B

## 15.3 Recursion

In LOTOS no explicit recursive operator is provided. Recursive processes can be defined by means of recursive process definitions. As we have already done for the relabelling operator, we prefer to use an explicit recursive operator to give the related laws [34]. Notation: rec x.E denotes a recursive process; E[rec x.E/x] denotes the process resulting by substituting in E every occurrence of x with rec x.e.

rec x. E = E[rec x.E/x]

If F = E[F/x] then F =rec x.E , provided that x is guarded in E (see section 15.4.1)

Note that the addition of these two laws for recursive processes to the ones given before, does not provide a complete set of laws for observational congruence for basic LOTOS.

## 15.4 Laws for reducing unguarded recursions to guarded recursions

In the following we give the transformation rules that allow to reduce unguarded recursion for the language composed by the action prefix, (explicit) recursion, choice and pure synchronization, preserving the trace congruence.

### 15.4.1 Definitions

An occurrence of a process variable x in a behaviour expression E is *guarded* in E if it occurs within some subexpression a;F of E, with $a \in Act$. Otherwise it is said to be *unguarded* in E [34].

A recursive process definition rec x.E is said to contain a *guarded recursion* if the process variable x occurs guarded in E. Conversely, it is said to contain an *unguarded recursion* if x occurs unguarded in E.

### 15.4.2 Unguarded Recursion Laws

The following laws, in conjunction with the trace congruence and the guarded recursion ones, form a complete set of axioms for the trace congruence on the LOTOS subset composed by the action prefix, recursion, choice and pure synchronization, allowing unguarded recursion. [20].

rec x. (U(x)[]B(x)) = rec x. B(x)

rec x. ((U(x)[]B1(x)) || B2(x)) = rec x. (B1(x) || B2(x))

where B(x), B1(x), B2(x) are generic behaviour expressions, and U(x) is an unguarded expression of the form x or x || B(x)

Useful instances of the above axioms are:

rec x. x = stop

rec x. (x [] B(x)) = rec x. B(x)

rec x. (x || B(x)) = stop

The second of these derived laws has been introduced in [34] for the language without synchronization, and actually preserve strong congruence; the more general laws handling synchronization does not in general preserve strong congruence, due to their interactions with the trace congruence law B || B = B .

### 15.4.3 More Unguarded Recursion Laws

Another useful transformation, operating on a subset of LOTOS including the interleaving operator, is the following:

rec x. $[]\{a_i; Q_i\}|||x$ = rec x. $[]\{a_i; Q_i|||x\}$

This law preserves strong equivalence, as presented in Chapter 9.