Equivalence relations
(*This annex is not an integral part of the body of this International Standard.*)

B.1 Introduction

The formal model of LOTOS provides a rigorous mathematical basis for the analysis of LOTOS specifications. This basis enables the properties of specifications to be studied by using only relevant mathematical theory, provided that it can be applied to this formal model.

Algebraic transformation laws enable subexpressions of expressions to be replaced by equivalent expressions without change of meaning. Such laws simplify the analysis of algebraic specifications. In the analysis of LOTOS specifications this approach can be applied both to the behaviour expressions and value expressions (ACT ONE terms). In this annex only the transformation laws for behaviour expressions will be considered. More work on the verification of behaviour expressions, and work on the verification of abstract data types can be found in the references of this annex.

Using algebraic laws behaviour expressions may be transformed into further equivalent expressions by repeated substitution. This method may be applied to demonstrate that two descriptions of a system are equivalent. For example one description might be the original specification and the other a description in LOTOS of the implementation. The other description might also be a more explicit description of the behaviour of the systems of which some properties can be more readily verified.

In clauses B.2 and B.3 transformation laws are given for LOTOS behaviour expressions. These laws are attained by identifying labelled transition systems, being the formal interpretation of closed behaviour expressions, following certain equivalence relations that are defined over such transition systems. The equivalence relations used in clauses B.2 and B.3 both relate transition systems that have indistinguishable observable behaviour, i.e. by observing (=interacting with) such systems an experimentor is unable to distinguish one system from an equivalent one.

The equivalence relations used in clauses B.2 and B.3 are termed 'weak bisimulation equivalence' and 'testing equivalence', respectively. They are different in the way in which the concept of 'observable equivalence' is formalized. Two expressions that are 'weak bisimulation equivalent' are also 'testing equivalent', but not vice versa. In other words, the formalization of 'bisimulation' distinguishes between more systems than 'testing'. The choice between these two concepts of equivalence must be made by those who wish to use the transformation laws, depending on which relation reflects their technical criteria for identification best.

Unfortunately, neither the weak bisimulation relation nor the testing equivalence possess the substitution property on LOTOS behaviour expressions described above in all contexts. When considered as systems in isolation there are behaviour expressions that are equivalent, but which will behave differently when they are part of a larger system. This may occur, for example, if they form a component of a choice expression.

Therefore, stronger relations between behaviour expressions are required that do have the full substitution property. Such relations are termed congruence relations. Clauses B.2.2 and B.3.2

contain the laws for the bisimulation congruence relation, and the testing congruence relation, respectively.
The sets of laws that are presented in the subsequent parts of this annex are not 'complete', i.e. if two expressions are in a certain equivalence/congruence relation, this does not imply that one expression can be transformed into the other using the relevant set of laws. Of course, the converse implication does hold: the sets of laws are 'sound'.

B.2 Weak bisimulation

B.2.1 Definitions

Let Sys = <*S*,*A*,*T*,s_0>be a labelled transition system.

Let s,*s*'∈*S*, a_1,..,a_n∈*A*,then the following notation is defined:

s-;a_1,..,a_n→*s*' iff there exist s_1,..,s_{n+1} ∈ S withs =s_1,s' =s_{n+1}, and s_i−a_i→s_i+1 for all $1 \le i \le n$

Lets,*s*'∈ *S*, *a* ∈ *A*,*k* be a natural number then the following notation is defined:

s−a^k→ *s*' iff *s*-*a*,*a*,...,*a* → *s*' where *a*,*a*,...,*a* is a list of *k* occurrences of *a*

Let *s*, *s*' ∈ *S*,a_1,..,a_n ∈ *A*-{i}, then the following notation is defined:

s = a_1,..,a_n ⇒ *s*' iff there exist natural numbers k_0,..,k_n with *s* − i^{k0}, a_1, i^{k1},..., a_n, i^{kn} → *s*'

s ε⇒ *s*' iff either *s* = *s*'or there exists a natural number *n* with *s* - i^n → *s*'

Let *S* be a set of states. A relation *R* ⊆ *S* × *S* is a *weak bisimulation relation* iff:
If < S_1 ,S_2 > ∈ *R* then for all *t* ∈ (*A*-{i})*

- a) there exists an s_1'∈*S* with s_1 =*t* ⇒ s_1' implies there exists ans_2'∈ *S* withs_2 = *t* ⇒s_2 'and < S_1',S_2' > ∈*R*;
- b) there exists an s_2'∈*S* withs_2 =*t* ⇒ s_2'implies there exists an s_1' ∈ *S* with s_1 = *t*⇒ s_1 ' and < s_1',s_2' > ∈ *R* ;

Let *Sys*$_1$ and *Sys*$_2$ be labelled transition systems, with state sets S_1 and S_2, and initial states s_{01} and s_{02}, respectively. *Sys*$_1$ and *Sys*$_2$ are *weak bisimulation equivalent* iff there exists a weak bisimulation relation *R* ⊆ *S* ×*S*, with

- a) *s* = S_1 ∪ S_2;
- b) < s_{01},s_{02} > ∈ *R*

Two closed LOTOS behaviour expressions, in the context of an algebraic specification and a complete process environment, are weak bisimulation equivalent iff their transition systems as defined in 7.5.4. are weak bisimulation equivalent.
Two LOTOS behaviour expressions B_1,B_2 with all their free value-identifiers contained in {x_1,..,x_n} are weak bisimulation equivalent, iff all instances [E_1/x_1,..E$_n$/x$_n$]B_1 and [E_1/x_1,..,E_n/x_n]B_2 are weak

bisimulation equivalent, where E_1,...,E_n are closed value expressions of the same sort as x_1,..,x_n, respectively.
A LOTOS *context C* [] is a LOTOS behaviour expression with a formal process parameter ' [ ] '. If *C* [ ] is a context and *B* is a behaviour expression then*C* [*B*] is the behaviour expression that is the result of replacing all ' [ ] ' occurrences in *C* [ ] by *B*.
Two LOTOS behaviour expressions B_1, B_2 are *weak bisimulation congruent* iff for all LOTOS contexts *C*[], *C* [B_1] is weak bisimulation equivalent with *C* [*B*2].

B.2.2 Laws for weak bisimulation congruence.

If two LOTOS behaviour expressions B_1 and B_2 are weak bisimulation congruent this is written $B_1 = B_2$, where the use of ' = ' is justified by the fact that B_1 may be replaced by B_2 and vice versa in all LOTOS contexts without changing the meaning. Equations of the form $B_1 = B_2$ involving variables for behaviour expressions are called (weak bisimulation congruence) *laws*. The most useful laws for weak bisimulation congruence are listed below. The formulation of some of these laws depends on the *label sets* of some of the involved behaviour expressions. The label set *L*(*B*) of a behaviour expression *B* is defined as the set of all gate identifiers that occur in *B* that are not bound by a hiding-operator, or a sum-domain- or par-domain expression.

- a) *Action-prefix*
  Let *g*..? *x* : *t*.. be an action-denotation with an experiment-offer ? *x* : *t*, and let *z* be a value-identifier that does not occur in *g* ... ? *x* : *t* ... [*E*]; *B*.
  - 1) *g* ... ? *x* : *t* ... [*E*]; *B* = *g* ... ? *z* : *t* ... [*z* / *x*][*E*];[*z*/*x*] *B*
  - 2) *g* ... ? *x* : *t* ... ;*B* = **choice** *x* : *t* [] *g* ... ! *x* ... ; *B*
  - 3) *g* ! E_1 ... !E_n [*E*]; *B* = [*E*] -> *g* ! E_1 ... ! E_n; *B*
- b) *Choice*
  1) B_1 [] B_2 = B_2 [] B_1
  2) B_1 [] (B_2 [] B_3) = (B_1 [] B_2) [] B_3
  3) *B* [] **Stop** = *B*
  4) *B* [] *B* = *B*
  5) [*E*/*x*]*B* [] (**choice** *x*:*t* [] *B*) ≈ **choice** *x*:*t* [] *B* if [*E*] ∈ *Q*(*t*)
  6) **choice** *x*:*t*[]*B* = *B* if *x* is not free in *B*
  7) **choice** *x*:*t*[] **exit**(... ,*x*, ...) = **exit**(... ,**any** *t*, ...)
- c) *Parallel*
  If a law holds for all of the parallel operators '|' is used to denote any of them (using the same instance throughout the law)
  1) B_1 | B_2 = B_2 | B_1
  2) B_1 |(B_2 | B_3) = (B_1 | B_2)| B_3
  3a) **exit**(E_1, ... ,E_n) | **exit**(E_1', ... ,E_n') = **exit**(E_1, ... ,E_n) if *n* = *m* and ([E_i] = [E_i']or (E_i' = **any** *t* and *sort*(E_i) = *t*)) for all *i*'with 1 ≤ *i* ≤ *n*
  = **stop** otherwise

3b) **exit**(...) | **stop** = **stop**

Let *A*,*A*' be lists of gate-identifiers

4) B_1 |[*A*]| B_2 = B_1 |[*A* ']|B_2 where *A* ' is any list containing the same elements as A

5) B_1 |[*A*]| B_2 = B_1 |[*A* ']| B_2 where $A' = A \cap (L(B_1) \cup L(B_2))$

6) B_1 |[*A*]|B_2 = B_1 || B_2 if $(L(B_1) \cup L(B_2)) \subseteq A$

7) B_1|[ ]|B_2 = B_1 ||| B_2

- d) *Enabling*

  Let >>* denote any instance of the enable operator.

  - 1) **Stop**>>* *B* = **Stop**
  - 2a) **exit** >>*B* = **i** ; B
  - 2b) **exit**(E_1, ... ,E_n) >> **accept** x_1:t_1, ... ,x_n:t_n **in** *B* = **i** ; [E_1/x_1, ... ,E_n/x_n]*B*
  - 3) (B_1 >> B_2) >> B_3 = B_1 >> (B_2 >> B_3)
  - 4) *B* >>* **stop** = *B* ||| **stop**

- e) *Disabling*

  - 1) B_1 [>(B_2 [> B_3) = (B_1 [> B_2)[> B_3
  - 2) *B*[> **stop** = *B*
  - 3) (B_1 [> B_2)[ ] B_2 = B_1 [> B_2
  - 4) **stop**[> *B* = *B*
  - 5) **exit**(...)[> *B* = **exit**(...)[ ]*B*

- f) *Hiding*

  Let *A*,*A*',*A*" be lists of gate-identifiers, >>* any instance of the enable-operator

1) **hide** *A* **in** *B* = **hide** *A* ' **in** *B* **if *A* ' is a list containing the same elements as A**
2) **hide** *A* in *B* = **hide** *A* ' **in** *B* if *A*' = $A \cap L(B)$
3) **hide** *A* **in hide** *A* ' **in** *B* = **hide** *A*" in *B* where *A*" ≈ $A \cup A'$
4) **hide** *A* **in** *B* = *B* if $A \cap L(B) = \phi$
5a) **hide** *A* **in** *a*!E_1 ... !E_n;*B* = **i**;(**hide** *A* **in** *B*) if $a \in A$
5b) **hide** *A* **in** *g*;*B* = *g*;(**hide** *A* **in** *B*) if name(*g*) $\notin A$
6) **hide** *A* **in** B_1 [] B_2 = (**hide** *A* **in** B_1) [ ] (**hide** *A* **in** B_2)
7) **hide** *A* **in** (B_1 |[*A* ']| B_2) = (**hide** *A* **in** B_1) |[*A* ']| (**hide** *A* **in** B_2) if $A \cap A' = \phi$
8) **hide** *A* **in** (B_1 >>* B_2) = (**hide** *A* **in** B_1) >>* (**hide** *A* **in** B_2)
9) **hide** *A* **in** (B_1 [> B_2) = (**hide** *A* **in** B_1) [> (**hide** *A* **in** *Bz)*
10) **hide** *A* **in**[*E*]-> *B* = [*E*]->(**hide** *A* **in** *B*)

- g) *Guarding*

  1a) [*L* = *R*]-> *B* = *B* if *L* = *R*

  [*L* = *R*]-> *B* = **stop** otherwise

  1b) [*BE*] -> *B* = [*BE* = true]-> *B* if *BE* is a value-expression

- h) *Instantiation*

  b[a_1, ... ,a_n](E_1, ... ,E_m) = ([E_1/x_1, ... ,E_m/x_m]B_b)[a_1/g_1, ... ,a_n/g_n]

  if **process** *b* [g_1, ... ,g_n](x_1, ... ,x_m):*f* : = B_b **endproc** is the format of the corresponding process abstraction for the process-identifier *b*.

- j) *Local definition*

  let x_1:t_1 = E_1, ... ,x_n:t_n = E_n **in** *B* = [E_1/x_1, ... ,E_n/x_n]*B*

- k) *Relabelling*

  Let [S] be any instance [a_1/g_1, ... ,a_n/g_n] of the (post-fix) relabelling operator. We associate with [S] the function S on gate-identifiers defined by

  $S(g_i) = a_i$ $(1 \le i \le n)$

  $S(g) = g$ if $g \neq g_i$ $(1 \le i \le n)$

  S can be extended to lists, sets, strings etc. containing gate-identifiers, in which case the application of S yields an object of the same category in which all occurrences g have been replaced by $S(g)$ for all gate-identifiers g.

  1) **stop**[S] = **stop**
  2) **exit**(...)[S] = **exit**(...)
  3) (a;B)[S] = $S(a)$;B[S]
  4) (B_1 [ ] B_2)[S] = B_1[S] [ ] B_2[S]
  5) (B_1 |[A]| B_2)[S] = B_1[S] |[$S(A)$]| B_2[S] if S is injective on $L(B_1) \cup L(B_2) \cup A$
  6) (B_1 >>* B_2)[S] = B_1[S] >>* B_2[S]
  7) (B_1 [> B_2)[S] = B_1[S] [> B_2[S]
  8) (**hide** A ' **in** B)[S] = **hide** A **in** B [S] if S is injective on $L(B) \cup A'$, and $S(A') = A$
  9) B[S] = B if S is the identity on $L(S)$
  10) B[S_1] = B[S_2] if $S_1(a) = S_2(a)$ for all $a \in L(B)$
  11) B[S_1][S_2] = B[$S_2 \cdot S_1$]

- m) *Internal action*
  - 1) a;**i**;B = a;B
  - 2) B[]**i**;B = **i**;B
  - 3) a;(B_1 [] **i**;B_2) [] a;B_2 = a;(B_1 [] **i**;B_2)
  - 4) [E/x]B[](**choice** x:t[]**i**;B) = **choice** x:t**[]i**;B if [E] ∈ $Q(t)$

- n) *Expansion theorems*

  For the sake of convenience, the following notational convention is introduced:

  B_1 [] B_2 [] ... [] B_n is written []{B_1, ... ,B_n}
  choice x:t [] B is written []{[E/x]B | [E] ∈ Q(t)}
  choice g **in** [a_1, ... ,a_n] [] B is written []{B [a_i/g] | $a_1 \in \{a_1, ... ,a_n\}$)

  In this way we have introduced the general format []S where S is a set of behaviour expressions. It is assumed that the elements of S can always be enumerated by some suitably chosen index set. If the elements of S are all of the form $b_i;B_i$ then the following useful laws apply

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

  1) B|[A]| C =
  []{b_i;(B_i,|[A]|C) | *name*(b_i)∉ A, $i \in I$}
  [] []{c_j;(B|[A]|C_j) | *name*(c_j)∉ A, $j \in J$]
  [] []{a;(B_i|[A]|C_j) | $a = b_i = c_j$, *name*(a)∈A, $i \in I$, $j \in J$}
  if all b_i ($i \in I$), C_j($j \in J$) are of the form $g!E_1$ ... $!E_n$

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

  3) hide A **in** B =
  []{b_1;**hide** A **in** B_i | *name*(b_i)∈A, $i \in I$}

[] []{**i**;**hide** *A* **in** B_i| *name*(b_i)∈*A*, *i*∈*l*}
if all b_i (*i*∈*l*) are of the form *g*!E_1 ... !E_n

4) *B*[*S*] = []{*S*(b_i);B_i[*S*] | *i* ∈ *l* }

B.2.3 Laws for weak bisimulation equivalence

B.2.3.1 Notation

If two behaviour expression B_1 and B_2 are weak bisimulation equivalent this is denoted by $B_1 = B_2$.

B.2.3.2 General law

B ≈ **i**; *B*

B.2.3.3 Rules for ≈

- 1) Let*C* [ ] be a LOTOS context of the following forms:
  - a) *g*;[ ]
  - b) [ ]|[*A*]|*B* , or *B*|[*A*]|[ ]
  - c) [ ]>>* *B*, or *B*>>*[ ]
  - d) [ ] [> *B*
  - e) **hide** *A* **in** [ ]
  - f) [*E*]->[ ]
  - g) [ ][*S*]
  - h) **let** ... **in** [ ]

  then if $B_1 \approx B_2$ then *C* [B_1] ≈ *C* [B_2]
- 2) if *B* ≈ *C* then *a*;*B* = *a*;*C* for all action-denotations '*a*'
- 3) if *B* = *C* then *B* ≈ *C*

B.3 Testing equivalence

B.3.1 Definitions

Let *Sys* = < *S*,*A*,*T*,s_0 > be a labelled transition system.

Let *S* ∈ *S*, then the set (*s* **after** *t*) ⊆ *S*, for *t* ∈ (*A*-{**i**})* is defined by *s* **after** *t* = {*s*'| *s* = *t*⇒ *s*'}
Let *L* ⊆ *A*-{**i**}, then the predicate (*s* **must** *L*) is defined by *s* **must** *L* iff *s* =ε⇒*s*' implies that there exists an *s*'' with *s*' =*a*⇒ *s*'' for some *a* ∈ *L*
Let *R* ⊆ *S*, then the predicate (*R* **must** *L*) is defined by *R* **must** *L* iff *s* **must** *L* for all *s* ∈ *R*
Let Sys_1 = < S_1,A_1,T_1,s_{01} > and Sys_2 = <S_2,A_2,T_2,s_{02} >, and extend them to the label set *A* = A_1 ∪ A_2, then the predicate (Sys_1 red Sys_2) is defined by

Sys_1 red Sys_2 iff

(S_{02} **after** *t*) **must** *L* implies (s_{01} **after** *t*) **must** *L* for every *t* ∈ (*A*-{**i**})* for every *L* ⊆ (*A*-{**i**})

For closed LOTOS behaviour expressions SB_1, B_2, in the context of an algebraic specification and a

complete process environment, B_1 **red** B_2 iff for their transition systems Sys_1, Sys_2 respectively, as defined in 7.5.4, Sys_1**red** Sys_2.

For LOTOS behaviour expressions B_1,B_2 with all their free value-identifiers contained in $\{x_1, ... ,x_n\}$ B_1 **red** B_2 iff

[E_1/x_1, ... E_n/x_n]B_1 **red** [E_1/x_1, ..., E_n/x_n]B_2 for all E_1, ... ,E_n, where E_1, ... ,E_n are closed value expressions of the same sort as x_1, ..., x_n, respectively.

Two LOTOS behaviour expressions B_1,B_2 are *testing equivalent* iff B_1 **red** B_2 and B_2 **red** B_1.

For LOTOS behaviour expressions B_1,B_2 B_1 **cred** B_2 iff for all LOTOS contexts$C$ [ ] C [B_1] **red** C [B_2].

Two LOTOS behaviour expressions B_1,B_2 are *testing congruent* iff B_1 **cred** B_2 and B_2 **cred** B_1.