

© British Standards Institution. No part of this publication may be photocopied or otherwise reproduced without the prior permission in writing of BSI

Draft for Development

Open Systems Interconnection: LOTOS, a formal description technique based on the temporal ordering of observational behaviour (ISO/DP 8807)

To avoid delay in preparing the content of this publication for printing, it was reproduced from the only available copy of the finally agreed document. The resultant poor print quality is very much regretted. The price has been adjusted accordingly.



Documents

Document Sharing Hub

Contents

	BS page		BS page
National foreword	2	5.2 CCS*	9
Committees responsible	Back cover	5.3 Semantics of CCS*	15
Draft for Development		6 Formal syntax	20
0 Introduction	3	6.1 Lexical tokens	20
0.1 General	3	6.2 Specification text	23
0.2 FDTs	3	7 Semantics	27
0.3 The requirement for standard FDTs	3	7.1 Introduction	27
0.4 The objectives to be satisfied by an FDT	3	7.2 The static semantics	27
0.5 The origin of LOTOS	4	7.3 The dynamic semantics	35
0.6 Act one	4	Annex	
0.7 The structure of this document	4	A Verification	41
1 Scope and field of application	5	National Appendix	
2 References	5	A British Standards and Drafts for Development for OSI, data communications and computer graphics	44
3 Definitions	5	Tables	
4 Basic mathematical concepts and notation	6	1 Metalanguage symbols	7
4.1 General	6	2 CCS* syntax table	13
4.2 Sets	6		
4.2 Lists (see national foreword)	6		
4.3 Relations and functions	7		
4.4 Backus-Naur Form	7		
5 Model	9		
5.1 Processes and interaction	9		

This publication is not to be regarded as a British Standard.

It is being issued in the Draft for Development series of publications and is of a provisional nature because the subject matter is in the process of ongoing development within International Standards Committee (ISO/TC 97/SC 21 Information retrieval, transfer and management for Open Systems Interconnection).

It should be applied on this provisional basis, so that information and experience of its practical application may be obtained.

A review of this Draft for Development will be carried out not later than two years after its publication. Notification of the start of the review period, with a request for the

submission of comments from users of this Draft for Development, will be made in an announcement in the appropriate issue of *BSI News*. According to the replies received, the responsible BSI committee will judge whether the Draft for Development can be converted into a British Standard or what other action should be taken.

Observations which it is felt should receive attention before the official call for comments will be welcomed. These should be sent to the Joint Secretaries of BSI Technical Committee OIS/121 at the BSI Head Office address as shown on the back cover.

National foreword

This Draft for Development has been prepared under the direction of the Office and Information Standards Committee. The primary objective of this Draft for Development is to give advance guidance to product and systems designers concerned with formal description techniques in the Basic Reference Model of Open Systems Interconnection (OSI). The text of this Draft for Development is identical with ISO/DP 8807 which is issued by the International Organization for Standardization (ISO).

For ease of reproduction the ISO text has been reproduced unchanged.

The UK is actively participating through BSI Technical Committee OIS/121 in the development of the ISO Draft Proposal. The Technical Committee will review any agreed amendment or addenda to the ISO text and will, where appropriate, update the text of this Draft for Development by means of an amendment or revision. When the ISO draft reaches an appropriate level of agreement and stability, this Draft for Development may then be converted to a British Standard. For information, a list of published British Standards and Drafts for Development (together with those scheduled for publication at the time of publication of this Draft for Development) for OSI, data communications and computer graphics is attached as national appendix A. Press releases for new OSI, data communications and computer graphics British Standards publications will be issued to the computing press, and will be recorded in full in *BSI News*.

Terminology and conventions. The text of the Draft Proposal has been approved as suitable for publication as a Draft for Development without deviation. Some terminology and certain conventions are not identical with those used in Drafts for Development; attention is drawn especially to the following.

Wherever the words 'International Standard' appear, referring to this Draft for Development, they should be read as 'Draft for Development'.

Wherever page numbers are quoted, they are ISO page numbers.

Cross-reference

International Standard	Corresponding British Standard
ISO 7498-1984	BS 6568 : 1984 Description of basic reference model for open systems interconnection (Identical)

NOTE. *Textual error.* Clause 4.2 Lists should be read as 4.3. Clauses 4.3 and 4.4 should be read as 4.4 and 4.5 respectively.

0. Introduction

0.1 General

Formal description techniques (FDTs) are methods of defining the behaviour of an (information processing) system without using a natural language such as English. In the following sub-clauses of this introduction, the importance of FDTs and their standardization is discussed. The objectives that an FDT must satisfy are considered. The origin of LOTOS is discussed. Finally the structure of this document is explained.

0.2 FDTs

Formal description techniques are important tools for the design, analysis and specification of information processing systems. It is by means of formal techniques that system descriptions can be produced that are **complete, consistent, precise, concise** and **unambiguous**. This is only possible if an FDT is self-contained, so that the descriptions given in an FDT need not refer to any informal knowledge of the system that is described. An important aspect of a formal system is that it allows analysis by mathematical methods. An FDT that has such a formal, mathematical basis can be used to prove the correctness of a specification.

0.3 The requirement for standard FDTs

If an FDT is defined in an International Standard, the description is available to all who require it. The Directives for the production of such a standard require a high degree of international acceptance and technical stability. Any amendment also requires international agreement. Hence FDT a standard FDT offers the most useful form of presentation to those who wish to apply it.

0.4 The objectives to be satisfied by an FDT

Although this document describes an FDT that is generally applicable to distributed, concurrent information processing systems, it has been developed particularly for OSI. The main objectives to be satisfied by such an FDT are that it should be:

- (a) **expressive**: an FDT should be able to define both the protocol specifications and the service definitions of the seven layers of OSI described in ISO 7498.
- (b) **well-defined**: an FDT should have a formal mathematical model that is suitable for the verification of these specifications and definitions. The same model should support the validation of implementations that are permitted by the OSI International Standards. This model should also support the testing of an implementation for conformance.
- (c) **well-structured**: an FDT should offer means for structuring the description of a specification or definition in manner that is meaningful and intuitively pleasing. A good structure increases the readability, understandability flexibility, analysability and maintainability of system descriptions.
- (d) **abstract**: there are two aspects of abstraction that an FDT should offer:
 - 1. an FDT should be completely independent of methods of implementation, so that the technique itself does not provide any undue constraints on implementors.
 - 2. an FDT should offer the means of abstraction from irrelevant details with respect to the context at any point in a description.Abstraction can reduce the local complexity of system descriptions considerably. In the presence of a good structure, abstraction can help even further to reduce the complexity of descriptions.

DD 124:1985

0.5 The origin of LOTOS

LOTOS (Language of Temporal Ordering Specification) was developed by FDT experts from ISO/TC97/SC21/WG16-1/FDT/Subgroup C during the years 1981-1984. The basic idea that LOTOS developed from was that systems can be described by defining the relation between events in the externally observable behaviour of a system. The formal mathematical model of LOTOS is based on a modification of the Calculus of Communicating Systems (CCS). CCS was developed at the University of Edinburgh. CCS provides a powerful analytical technique for concurrent systems.

0.6 ACT ONE

The representation of data structures and value expressions in LOTOS is by means of the abstract data type language ACT ONE. ACT ONE was developed at the Technical University of Berlin.

The part of LOTOS dealing with the description of process behaviour is not dependent upon ACT ONE. Many well-defined languages for the description of data structures could, in principle, be used in combination with the process definition facilities of LOTOS.

EDITOR'S NOTE. A complete definition of ACT ONE is not included in this draft. Reference is made to the defining document [ACT ONE]. The definition of ACT ONE is expected to be included in this draft before it reaches the DIS stage. Collaboration with CCITT is currently leading to rapid progression of this work.

0.7 The structure of this document

This document differs in contents from most International Standards. The importance of a formal, mathematical basis of an FDT (see clauses 0.2 and 0.4(b)) makes the inclusion of mathematical material in this definition necessary. This introduces problems of presentation and style. The main body of mathematical material is presented in clauses 5 and 7 that deal with the formal semantics of LOTOS. It has been the aim to make these clauses as self-contained as possible, but a certain mathematical knowledge is required for their understanding. This part of the document, together with Annex A is important for the verification, validation and testing aspects of LOTOS descriptions.

Clauses 6 and 7.2 contain the part of the definition of LOTOS that explains how correct LOTOS specifications can be produced. This part of the document requires only knowledge of basic mathematical concepts. In clause 4 these concepts are introduced together with notation that is used throughout this document.

[ISO page 2]

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 1.

1. Scope and field of application

This document defines the semantics and syntax of the Formal Description Technique LOTOS. LOTOS is in general used for the formal description of distributed, concurrent information processing systems. In particular LOTOS can be used to formally describe the service definitions and protocol specifications of the layers of Open Systems Interconnection described in ISO 7498.

2. References

- | | |
|-----------|---|
| ISO 7498 | Information Processing Systems - Open Systems Interconnection - Basic Reference Model. |
| [Milner] | R.Milner, A Calculus of Communicating Systems, Springer-Verlag (Berlin 1980). |
| [ACT ONE] | H.Ehrig,W.Fey,H.Hansen, ACT ONE: an algebraic specification language with two levels of semantics, Bericht-Nr. 83-03, Technical University of Berlin (Berlin 1983). |

3. Definitions

Definitions which are related to basic mathematical concepts are given in clause 4. Definitions of other concepts are given at the point of their first occurrence in this document.

EDITOR'S NOTE. In this document definitions are given in the places where they have conceptual relevance, instead of in clause 3 where they are usually found. It is considered to use clause 3 for the definition of the most basic concepts that are used in this document, and that are generally assumed to be known to FDT experts. Examples of these most basic concepts are: model, syntax, semantics, natural number, etc.

[ISO page 3]

Documents

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 1.

DD 124:1985

4. Basic mathematical concepts and notation

This clause contains a list of basic mathematical concepts and related notations used in clauses 5, 6, and 7.

4.1 General

$\equiv$ is defined as.
 iff if and only if (i.e. double implication).

4.2 Sets

$\{a, b, c, \dots\}$ the **set** made up of elements $a, b, c, \dots$.
 $\{\}$ the **empty set** (i.e. the set having no elements).
 $x \in A$ x is an **element** of the set A .
 $x \notin A$ x is **not** an **element** of the set A .
 $A \subseteq B$ A is a **subset** of B (i.e. all elements of A are also element of B).
 $A \subset B$ A is a **proper subset** of B (i.e. $A \subseteq B$ and there is at least one element of B which is not an element of A).
 $A \cup B$ the **union** of A and B (i.e. the set which contains only all elements of A and all elements of B).
 $A \cap B$ the **intersection** of A and B (i.e. the set which contains only all elements of A which are also elements of B); sets A and B are **disjoint** iff $A \cap B = \{\}$.
 $A - B$ the **difference** of A and B (i.e. the set which contains only all elements of A which are **not** also elements of B).
 $A \times B$ the **Cartesian product** of A with B (i.e. the set of all ordered pairs, $\langle a, b \rangle$ such that $a \in A$ and $b \in B$).
 $P(A)$ the **powerset** of A (i.e. the set of all subsets of A).
 $\{x \in A | Q(x)\}$ the set which contains only all those elements of A which satisfy property Q (the abbreviation $\{x | Q(x)\}$ is used where set A may be deduced from the context).

4.2 Lists

$a_1, \dots, a_n$ the finite list (or sequence, or n -tuple) made up of the elements $a_1, \dots, a_n$.

[ISO page 4]

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 1.

Documents
Document Sharing Hub

4.3 Relations and functions

 $R \subseteq A \times B$

R is a **binary relation** between A and B (i.e. a set of elements of $A \times B$);
 the **domain** of R is defined as $\{a \in A \mid \text{there exists some } b \in B \text{ such that } \langle a, b \rangle \in R\}$;
 the **range** of R is defined as $\{b \in B \mid \text{there exists some } a \in A \text{ such that } \langle a, b \rangle \in R\}$.

 $f: A \rightarrow B$

f is a (**partial**) **function** from A to B (i.e. f is a binary relation between A and B such that for each $a \in A$, there exists at most one $b \in B$ such that $\langle a, b \rangle \in f$);
 if such an element $\langle a, b \rangle$ exists, then f is **defined** for a ;
 if $\langle a, b \rangle \in f$, then $f(a) = b$;
 the function f is **total** iff it is defined for all $a \in A$;

a function $f: A \rightarrow B$ is **injective** iff, for all a_1, a_2 in the domain of f , $f(a_1) = f(a_2)$ implies that $a_1 = a_2$;

the **inverse function**, $f^{-1}: B \rightarrow A$, of an injective function is defined as follows: for $b \in B$, if there exists an $a \in A$ such that $f(a) = b$, then $f^{-1}(b) = a$, otherwise f^{-1} is not defined.

4.4 Backus-Naur Form

The metalanguage used in this document to specify the syntax of the constructs is based on Backus-Naur Form. Table 1 lists the meanings of the various metasymbols.

Table 1 Metalanguage symbols

Meta-symbol	Meaning
=	is defined to be
	alternatively
.	end of definition
[x]	0 or 1 instances of x
{x}	0 or more instances of x
(x)	textual grouping
"xyz"	the terminal symbol xyz
meta-identifier	a non-terminal symbol

A meta-identifier is a sequence of letters and hyphens beginning with a letter.

A **production (rule)** is a meta-identifier, followed by '=', followed by a non-empty sequence of expressions containing terminal and/or non-terminal symbols, as defined by table 1.

A sequence of expressions containing terminal and non-terminal symbols in a production implies the concatenation of the text that they ultimately represent. The nature of this concatenation is subject to the rules given in 6.1.

Use of the words **of**, **in**, **containing** and **closest-containing** when expressing a relationship between terminal or non-terminal symbols has the following meanings:

- (a) the x **of** a y: refers to the x occurring directly in a production defining y.
- (b) the x **in** a y: is synonymous with 'the x of a y'.
- (c) a y **containing** an x: refers to any y in the defining production of which either x occurs directly, or a z containing an x occurs directly.
- (d) the y **closest-containing** an x: refers to that y which contains an x but does not contain another y containing that x.

5. Model

5.1 Processes and interaction

A **process** is an abstract entity that communicates with other processes in its environment. Communication between processes takes place at **interaction points**. An interaction point is an abstract resource that is shared by processes. Communication between processes is in the form of primitive interactions or **events**. An event occurs always between two (or possibly more) processes that share an interaction point. An event is an **atomic, synchronized** interaction between two such processes. An event is atomic because it cannot be substructured into subevents and because events cannot overlap in time. An event is a synchronized interaction because the two communicating processes are simultaneously involved in its occurrence.

NOTE. This limited model of communicating processes can be used to construct more general forms of interaction. Interaction between two processes that is non-atomic (i.e. the interaction is not instantaneous and can be interrupted by or can overlap with other interactions) can be modelled by one or more sequences of events. Asynchronous interaction between processes, i.e. two processes are not simultaneously involved in the interaction (e.g. the action of sending data does not coincide with their reception), can be modelled by representing the communication medium causing the time difference as a separate process. This medium process itself can interact synchronously with the two communicating processes, whereas the asynchronous behaviour is represented by the time difference between the events in the interactions with the one process and with the other process.

5.2 CCS*

A CCS* model consists of non-empty sets of:

- (a) sort-identifiers (see 5.2.1.2, 5.2.1.3)
- (b) sorts (see 5.2.1.2, 5.2.1.3)
- (c) value-identifiers (see 5.2.1.3)
- (d) value-expressions (see 5.2.1.5)
- (e) label-identifiers (see 5.2.2.1)
- (f) behaviour-identifiers (see 5.2.2.2)
- (g) process-abstractions (see 5.2.2.2)

The sets of sort-identifiers, value-identifiers, label-identifiers, and behaviour-identifiers are pairwise disjoint.

In the CCS* model a unique naming convention is adopted. This means that two occurrences of the same identifier always refer to the same object.

NOTE. CCS* is the modification of CCS mentioned in 0.5 that is used for the mathematical formalisation of 5.1. The CCS* model which is defined in 5.2. and 5.3 is used in 7.3 to give the semantics of LOTOS.

DD 124:1985

5.2.1 Data types and value expressions

5.2.1.1 General

In the CCS* model no explicit definition is given for the representation of data-values. In clause 5.2.1 only a number of concepts and requirements are defined. These guarantee the consistency of the dynamic aspects of the theory of CCS* that deal with process behaviour, and the static aspects of CCS* that deal with the data. Any representation of the static aspects that can model the concepts and meet the requirements defined here is acceptable as a choice for this CCS* model.

NOTE. In LOTOS the choice has been made to use only ACT ONE in the definition of types and their associated sorts and value expressions. Since the CCS* model permits some freedom in the choice of type definition mechanism, ACT ONE may also be used for type definition in that model.

5.2.1.2 Sorts

A **sort** is a set of data-values. A CCS*-model has a finite number of sorts. Two different sorts have no elements in common.

Each sort has a name. This name is the **sort-identifier**. Different sorts have different sort-identifiers.

5.2.1.3 Standard sorts

There exists a standard sort **bool** = {true,false}.

5.2.1.4 Value-identifiers

Value-identifiers (also known as **variables**) are used to indicate (unknown) data-values. Each variable can only range over the data-values of one sort. This sort is called the sort of the variable. For each variable **x** its associated sort **sort(x)** is known.

NOTE. The nature of variables in both CCS* and LOTOS is different from that in imperative programming languages which allow assignment. A variable in CCS* is a mathematical variable, i.e. once a value is tied to a particular variable each occurrence of that variable refers to that value. In other words: a variable cannot be re-assigned. In this respect there is a similarity to applicative programming languages.

5.2.1.5 Value-expressions

Value-expressions are language expressions that describe data-values. Value-expressions may contain variables. The set of variables occurring in a value-expression **E** is denoted by **FV(E)** (see 5.2.2.5). **Closed** value-expressions are value-expressions that do not contain variables. Therefore, closed value-expressions indicate a particular, known data-value.

Each value-expression has a sort. The sort of a value-expression **E**, **sort(E)**, is the sort that contains the data-values described by **E**, **val(E)**.

For each sort and for each data-value **v** in that sort there exists a closed value-expression **E** with **val(E)=v**.

[ISO page 8]

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 1.

In this document a value-expression with value v is sometimes represented by v itself if there are no technical reasons to distinguish between values and their representation.

5.2.2 Description of processes in CCS*

5.2.2.1 Labels and relabelling

Labels are identifiers that are used for the representation of interaction points. The set of labels is **Lab**. **Lab** contains a special element d .

NOTE 1. d is the CCS* representation of successful termination in LOTOS (see 7.3.2(j)).

NOTE 2. The identifier L is used in this document for functions which yield a label or set of labels in **Lab**.

A **relabelling** S is a function $S: \text{Lab} \rightarrow \text{Lab}$.

5.2.2.2 Process abstraction

Behaviour-identifiers are identifiers that are used for the representation of processes.

A **process-abstraction** is of the form

$$p(x_1, \dots, x_n) := B$$

where p is a behaviour-identifier and B a behaviour-expression (see 5.2.2.5). In process-abstractions (mutual) recursion is allowed, i.e. process-instantiations (see 5.2.2.3) of all defined process-abstractions may occur in B .

For each distinct behaviour-identifier p that occurs in a CCS*-model the sort of p , $\text{sort}(p)$, and the label set of p , $L(p)$, are defined using the associated behaviour abstraction $p(x_1, \dots, x_n) := B$ as:

(a) $\text{sort}(p) == \text{sort}(x_1) \times \dots \times \text{sort}(x_n)$

(b) $L(p) == L(B)$ (see 5.2.2.5)

It is further required that $FV(B) \subseteq \{x_1, \dots, x_n\}$ (see 5.2.2.5).

5.2.2.3 Process instantiation

A **process-instantiation** is of the form

$$p(E_1, \dots, E_n)$$

where p is a behaviour-identifier and $E_1, \dots, E_n$ are value-expressions. For each process-instantiation $p(E_1, \dots, E_n)$, $\text{sort}(E_1) \times \dots \times \text{sort}(E_n)$ is equal to $\text{sort}(p)$.

DD 124:1985

5.2.2.4 Action denotation

An **action denotation** is either:

- (a) a label followed by a finite sequence of value-expressions (possibly zero), where each of the value expressions in the list are immediately preceded by an exclamation mark ('!'); or
- (b) 1.

Examples

```
label1!(3+8)!true!'string'
label2!11
i
label3
```

NOTE. An action denotation of the form **label!E₁...!E_n** has the following intuitive meaning. When it occurs in a behaviour expression (see 5.2.2.4) the process that is described by this behaviour expression can interact at the interaction point indicated by **label**, transferring the values of **E₁,...,E_n**. The action denotation **i** is known as the **internal event**.

5.2.2.5 Behaviour expressions

A behaviour expression is a CCS* expression that describes process behaviour. A behaviour expression is any expression that can be constructed by a finite application of the following rules:

- (a) **stop** is a behaviour expression.
- (b) if **a** is an action denotation and **B** is a behaviour expression, then **(a;B)** is a behaviour expression.
- (c) if **B** is a set of behaviour expressions then **ΣB** is a behaviour expression.
- (d) if **B₁** and **B₂** are behaviour expressions then **(B₁|B₂)** is a behaviour expression.
- (e) if **B₁** and **B₂** are behaviour expressions then **(B₁[>B₂])** is a behaviour expression.
- (f) if **B** is a behaviour expression and **A** is a subset of **Lab** then **(B\A)** is a behaviour expression.
- (g) if **B** is a behaviour expression and **S** is a relabelling then **(B[S])** is a behaviour expression.
- (h) a **process instantiation** is a behaviour expression.
- (i) if **B** is a behaviour expression and **E** is a value-expression with **sort(E)=bool**, then **([E]→B)** is a behaviour expression.

NOTE. The construction of CCS* behaviour-expressions is not given using a Backus-Naur Form grammar as defined in 3.2 since the actual representation of these expressions is not essential as long as there is a one-to-one correspondence between the representation chosen here and any alternative one.

For each behaviour expression **B** the set of free variables in **B**, **FV(B)**, and the set of external labels, **L(B)**, is defined. The construction of these sets for a given behaviour expression **B**, is explained by table 2. The table shows how a behaviour expression **B** may be built from behaviour expressions **B₁**, **B₂**, and **B'** with label sets **L(B₁)**, **L(B₂)** and **L(B')**. **B₁**, **B₂** and **B'** are referred to as the **direct subexpressions** of **B**. The operator which constructs **B** from **B₁**, **B₂** or **B'** is referred to as the **direct operator** of **B**.

[ISO page 10]

This publication is a Draft for Development and is therefore not to be regarded as a British Standard. See page 1.

Table 2 CCS* syntax table

SYNTAX TABLE FOR CCS* BEHAVIOUR EXPRESSIONS			
	B	L(B)	FV(B)
inaction	stop	{}	{}
summation	$\sum B$	$U\{L(B') \mid B' \in B\}$	$U\{FV(B') \mid B' \in B\}$
action-prefix	$(g! ; B')$ $(g!E_1 \dots !E_n ; B')$ $(i ; B')$	$L(B') \cup \{g\}$ $L(B') \cup \{g\}$ $L(B')$	$FV(B')$ $FV(B') \cup U\{FV(E_i) \mid 1 \leq i \leq n\}$ $FV(B')$
parallel-composition	$(B_1 \mid B_2)$	$L(B_1) \cup L(B_2)$	$FV(B_1) \cup FV(B_2)$
disabling	$(B_1 [> B_2)$	$L(B_1) \cup L(B_2)$	$FV(B_1) \cup FV(B_2)$
restriction	$(B' \backslash A)$	$L(B') \backslash A$	$FV(B')$
relabelling	$(B' [S])$	$S(L(B'))$	$FV(B')$
instantiation	$b(E_1, \dots, E_n)$	$L(b)$	$U\{FV(E_i) \mid 1 \leq i \leq n\}$
guarding	$([E] \rightarrow B')$	$L(B)$	$FV(B') \cup FV(E)$

table 5.1

A **priority** is defined between some of the operators. The operators in decreasing order of priority are:

restriction, relabelling (same priority)
 action-prefix
 guarding
 choice-composition (see 5.2.2.7)
 parallel-composition
 disabling

The following rules for omitting parentheses apply:

- if B is a behaviour expression, and B' is a direct subexpression of B, and the direct operator of B' has a higher priority than the direct operator of B then the enclosing parentheses of B' may be omitted.
- the outermost pair of parentheses of a behaviour expression that is not a sub-expression of another behaviour expression may be omitted.

Example

$a!x;p(x,y) \backslash \{a,b\} \mid b!y; \text{stop}$

is equivalent to

$((a!x;(p(x,y) \backslash \{a,b\}))) \mid (b!y; \text{stop}))$

[ISO page 11]

Documents

5.2.2.6 Substitution

A variable that occurs in a value-expression may be replaced by a value-expression, provided that the sort of the replacing value-expression is the same as the sort of the variable. A **substitution** is a finite number of such replacements carried out simultaneously.

Let $x_1, \dots, x_n$ be variables and $E_1, \dots, E_n$ value-expressions with $\text{sort}(x_i) = \text{sort}(E_i)$ ($1 \leq i \leq n$).

Let E be a value-expression. The substitution of $E_1, \dots, E_n$ for $x_1, \dots, x_n$ in E , $[E_1/x_1, \dots, E_n/x_n]E$, is defined as the value expression that is the result of the simultaneous replacement of all occurrences of $x_1, \dots, x_n$ in E by $E_1, \dots, E_n$ respectively.

Let B be a behaviour expression. The substitution of $E_1, \dots, E_n$ for $x_1, \dots, x_n$ in B , $[E_1/x_1, \dots, E_n/x_n]B$, is defined as the behaviour expression that is the result of replacing every value expression E occurring in B that is not a subexpression of a larger value-expression by $[E_1/x_1, \dots, E_n/x_n]E$.

Examples

- (a) $[(3+z)/x]a!x; b!y; p(x+1, y) = a!(3+z); b!y; p((3+z)+1, y)$
- (b) $[z/y, y/x]a!x; b!y; p(x+1, y) = a!y; b!z; p(y+1, z)$
- (c) $[z/y][y/x]a!x; b!y; p(x+1, y) = [z/y]a!y; b!y; p(y+1, y) = a!z; b!z; p(z+1, z)$

5.2.2.7 Extended behaviour expressions

An **extended behaviour expression** is any expression which is constructed by a finite application of the following rules:

- (a) each behaviour expression (5.2.2.5) is an extended behaviour expression,
- (b) each choice expression (see below) is an extended behaviour expression,
- (c) each extended action-prefix expression (see below) is an extended behaviour expression.

The **choice-composition** of two extended behaviour expressions B_1 and B_2 is the extended behaviour-expression $(B_1 [] B_2)$. If B_1 and B_2 are behaviour expressions then $(B_1 [] B_2)$ is $\Sigma\{B_1, B_2\}$.

An **experiment-offer** is either:

- (a) a value-expression, immediately preceded by an exclamation mark ('!'); or
- (b) a variable, immediately preceded by a question mark ('?').

Those of the latter form are referred to as **binding** experiment offers.

An **extended action denotation** is either:

- (a) i ; or
- (b) a label followed by a finite, possibly empty sequence of experiment-offers not containing more than one occurrence of the same binding experiment offer.