ISO/IEC JTC1/SC7
Software Engineering
Secretariat: CANADA (SCC)

ISO/IEC JTC1/SC7 N1995

1998-10-14

Document Type	Letter Ballot Summary
Title	Letter Ballot Summary of FCD 15437: Information technology - Enhancements to LOTOS (E-LOTOS).
Source	JCT1/SC7 Secretariat
Project	07.20.02.03
Status	Draft
References	N1994 (33N0188)
Action ID	FYI or ACT
Due Date	
Mailing Date	1998-10-14
Distribution	SC7_AG
Medium	Uuencoded Acrobat
No. of Pages	4
Note	Ballot was circulated as document 33N188

Address reply to: ISO/IEC JTC1/SC7 Secretariat
Bell Canada - IT Procurement & Supplier Quality
1050 Beaver Hall Hill, 2nd Floor, Montréal (Québec) Canada H2Z 1S4
Tel.: +1 (514) 391-8286 Fax: +1 (514) 870-2246
sc7@qc.bell.ca

ISO/IEC JTC1/SC7/N1995

LETTER BALLOT SUMMARY

Project:	07.20.02.03		
Subject:	Letter Ballot Summary of FCD 15437: Information technology - Enhancements to LOTOS (E-LOTOS).		
Reference:	N1994 (33N0188)	Ballot:	FCD 15437
Circulation date:	1998-06-03	Closing date:	1998-10-03
Circulated to:	SC7_AG, JTC1 Sec.	Circulated by:	JTC1/SC33 Secretariat

TABLE OF VOTING AND COMENTS RECEIVED

"P" members	**Approve**	**Disapprove**	**Abstain**	**Comments**	**Not voting**
Australia			X	X	
Belgium	X*				
Brazil			X		
Canada	X			X	
Czech Republic					X
Denmark					X
Finland	X				
France	X			X	
Germany					X
Hungary					X
Ireland					X
Israel					X
Italy	X				
Japan		X*		X*	
Korea					X
Mexico					X
Netherlands					X
Norway	X				
Romania					X
Russian Federation					X
Singapore					X
South Africa					X
Spain	X				
Sweden			X		
Thailand					X
UK			X	X	
Ukraine					X
USA					X

"O" and "L" members voting:

	Approve	Disapprove	Abstain	Comments	Not Voting
China					X
Estonia					X
Portugal					X
Liaison:					X

* Late Vote

AUSTRALIA

Lack of expertise

CANADA

Canada supports the E-LOTOS CD, which is high-quality and should be taken expeditiously to completion.

However Canada requests that 'inheritance' be included in E-LOTOS at the DIS stage, so that the language could be considered object-oriented.

Methods to include inheritance in LOTOS were discussed by Steve Rudkin in a well-known article appeared in FORTE IV (1992).

Canada believes that the inclusion of inheritance would considerably improve the usefulness of the language and its chances to be used in industrial applications.

FRANCE

(1) MAJOR COMMENT : It is regrettable that FCD 15437 does not support operator overloading, a feature present in all modern computer languages. Due to this lack, it will be impossible to write, e.g., simple arithmetic expressions in E-LOTOS using the usual syntax. AFNOR urges the E-LOTOS Committee to consider this issue, which is crucial for the applicability and success of E-LOTOS.

(2) MAJOR COMMENT : The existence of run-time type-checking (the \ etc " type and record subtyping) will make implementation complex and inefficient, for a negligible practical benefit, because similar can be introduced in a much simpler way. In this respect, it is unfortunate that the E-LOTOS Committee has not retained the lessons from LOTOS, not to introduce features in the language that are notoriously difficult to implement.

(3) TECHNICAL COMMENT : AFNOR suggests to simplify the definition of the generalized parallel operator following the technical proposal enclosed as an appendix.

JAPAN

JPN-001E (B Document Structure
Rationale:
The structure of this FCD document is not consistent with that of ISO International Standard documents.

Proposed Improvement:
The document should be re-written according to "ISO/IEC Directives,
Part3: Rules for the structure and drafting of International Standards".

JPN-002TL: Chapter 7 Predefined library
Rationale:
The predefined types are insufficient as compared with LOTOS, SDL, ML, and Z.

Proposed Improvement:
The predefined library should define the following types:

1.Representation of values decimal, hexadecimal, octal representation)
2.Power set
3.Tuple
4.Time

JPN-003E: Technical terms (4.3.3, 5.2.5, 5.5.2, and so on)
Rationale:
The FCD includes several technical terms which are neither defined nor narratively explained.

Proposed Improvement:
The FCD should briefly explain "signatures", "functor" (4.3.3), "diamond rule "(5.2.5), and "diamond importation" (5.5.2).

JPN-004TL (B Distinction between LOTOS and E-LOTOS
Rationale:
Some operators of E-LOTOS are written as the same symbols used in LOTOS, but their semantics are slightly different, because the semantics of E-LOTOS does not fully surpass that of LOTOS. Therefore, we are afraid that some readers may confuse the operators of E-LOTOS with those of LOTOS.

Proposed Improvement:
The FCD should annotate such differences explicitly. Also, we propose that it presents rewriting rules from the constructors of LOTOS to E-LOTOS, in order to illustrate how existing specifications written in LOTOS is translated into E-LOTOS. A short example of the rules is shown below:

LOTOS Operators	ELOTOS Operators
Stop	stop
exit(V)	-
exit	null
a ; B	a ; B
B1 [] B2	B1 [] B2
B1 || B2	B1 || B2
B1 |[G]| B2	B1 |[G]| B2
B1 ||| B2	B1 ||| B2
B1 >> v1,v2,...,vn in B2	B1 ; B2
B1 [> B2	B1 [> B2
PID[G](V)	PID[G](V)
choice v1,v2,...,vn [] B	choice v1,v2,...,vn [] B endch
par...	par ...

UNITED KINGDOM

Reason for abstention

The UK abstains on this ballot as it does not currently have resources available for this project

Comments

Despite the vote of abstention, the UK considers that SC 7 should take into account the following comments:

1. Title. The current title suggests that this standard consists of extensions in the form of an amendment to ISO 8807. We suggest that a title such as 'Information technology - Enhanced Lotos (E-LOTOS)' would be preferable.

2. Scope. The standard should have a Scope clause as required by the ISO Directives.

A Graphical Parallel Composition Operator for Process Algebras

Hubert Garavel and Mihaela Sighireanu

INRIA Rhône-Alpes and DYADE / VASY group
655, avenue de l'Europe
38330 Montbonnot St Martin
France
Tel: +(33) 4 76 61 52 24 Fax: +(33) 4 76 61 52 52
E-mail: hubert.garavel@inria.fr, mihaela.sighireanu@inria.fr
Web: http://www.inrialpes.fr/vasy

Abstract. Process algebras are suitable for describing networks of communicating processes. In most process algebras, the description of such networks is achieved using parallel composition operators. Noticing that the parallel composition operators commonly found in usual process algebras are often limited in expressiveness and/or non-intuitive for non-expert users, we propose a new parallel operator that allows networks of communicating processes to be described easily, in a simple and well-structured manner. We illustrate on various examples (token-ring network, client-server protocol, chessboard grid) the theoretical and practical merits of our operator.

1 Introduction

Process algebras have been designed as a theoretical framework for the study of concurrency. Classical examples of process algebras are: ACP [1], CCS [24, 25], CSP [15], MEIJE [7], etc. There also exist specification languages, which combine process algebraic concepts with features borrowed from (functional or imperative) programming languages. For example, the OCCAM [4] language based on CSP, the μCRL [12] language based on ACP, and the LOTOS [16] language which combines the best features of CSP and CCS.

Process algebras have undeniable advantages: expressiveness, compositionality, formal semantics given in terms of Labelled Transition Systems (LTS) [26] using structural operational semantics [27, 13], verification algorithms based on behavioural equivalences and preorders, refinement methods, etc. Process algebras have been used successfully many times to model the behaviour of real systems. In addition, simulators, model-checkers, and theorem-provers are available for analyzing process algebraic descriptions, e.g., [6, 5, 8].

In spite of these advantages, the classical process algebras also suffer from limitations in terms of usability (because of their steep learning curve, they often require a substantial training effort), readability (process algebraic descriptions

are sometimes difficult to understand), and coverage (important aspects of system description, such as timing, probabilistic aspects, and priorities, are not addressed, although such extensions have been proposed in the literature).

Fortunately, work is going on to extend and improve the existing process algebras. In particular, the International Standardization Organization (ISO) has been working since 1992 on the definition of a revised version of the LOTOS language. This revised version, named E-LOTOS and currently at the stage of Final Committee Draft [29], includes new features suitable for increasing both the expressiveness and user-friendliness of the language. The work on E-LOTOS has generated numerous proposals for enhancing both the data type part and the behaviour part of LOTOS (see e.g., [11] for an overview and a discussion on these issues).

In this paper, we focus our attention on the improvement of the parallel composition operators of LOTOS. Although we assume some basic knowledge of LOTOS, our proposals are generic enough for being applicable to other process algebras than LOTOS.

The paper is structured as follows. Section 2 introduces some basic definitions and notations. Section 3 suggests to replace the binary parallel composition operators found in most process algebras with a new n-ary operator, more suitable for an easy description of networks of communicating processes. Section 4 proposes further enhancements to this operator, by relaxing the *maximal cooperation* paradigm used in process algebras such as CSP or LOTOS. Section 5 illustrates the usefulness of this parallel operator on a concrete application, the ODP[1] trading function [17]. Finally, Section 6 gives some concluding remarks.

2 Basic definitions and notations

In the sequel, we use the following notations borrowed from the value-passing process algebras (especially, LOTOS) terminology.

We note $B_1, B_2, ...$ algebraic terms constructed using the classical behavioural operators (inaction, action prefix, choice, etc.); these terms are called *behaviours* or *processes*. For our purpose, an exact syntactic definition of behaviours is not required. We note "$B_1 = B_2$" the syntactic identity of terms B_1 and B_2.

We note $G_1, G_2, ...$ identifiers corresponding to communication channels; these identifiers are called *gates*. We define two particular gates: τ, which denotes a non-observable event, and δ, which is used to express the synchronized termination of concurrent processes. We note $\widehat{G_1}, \widehat{G_2}, ...$ lists (or sets) of gates.

We note $L_1, L_2, ...$ tuples of the form $\langle G, v_1, ..., v_n \rangle$, where G is a gate and $v_1, ..., v_n$ a (possibly empty) list of typed values; these tuples are called *actions* or *labels*. We note $gate(L)$ the gate corresponding to the first element of the tuple L.

Structural operational semantics defines how a behaviour is translated into a (possibly infinite) *labelled transition system* [26], which encodes all the possible

[1] Open Distributed Architecture

execution sequences of the behaviour. The possible evolutions of a behaviour are modelled by a transition relation noted "$B_1 \xrightarrow{L} B_2$", which expresses that B_1 can perform an action L and become B_2 afterwards.

3 From binary to n-ary parallel composition operators

In LOTOS and most process algebras, parallel composition operators play a crucial role in the description of concurrent systems. Basically, there are two main uses of parallel composition:

- In distributed systems and protocols, parallel composition is a natural mean to describe a set of distributed components that execute concurrently and communicate with each other by message passing: therefore, the operands of a parallel composition operator correspond to physically-distributed entities. In the taxonomy proposed by [32], this use of parallel composition is called the *resource-oriented specification style*.
- Parallel composition can also be used for refinement purpose. Typically, a given (sequential) component can be divided into a set of sub-components, each of which expresses temporal constraints on the occurrences of certain events. These sub-components are combined together using parallel composition, which acts as the logical conjunction of the corresponding temporal constraints, resulting in a constrained behaviour. In such case, parallel composition expresses neither physical distribution nor concurrency, but rather a logical modularization of the initial component. In the taxonomy of [32], this use of parallel composition is called the *constraint-oriented specification style*.

Because of this double use of parallel composition, we believe that a suitable parallel composition operator must support *multiway synchronization*, i.e., rendezvous synchronization involving more than two processes:

- As far as resource-oriented style is concerned, multiway synchronization is not really necessary: it is sufficient to describe the communication between an emitter and a receiver. Most process algebras allow such *handshaking* synchronization, with some differences with respect to the form of value exchanges that take place during the synchronization.
- But, as far as constraint-oriented style is concerned, multiway synchronization is mandatory. For instance, a controller for a robot with n degrees of freedom can be expressed as the parallel composition of n sub-processes, each sub-process controlling the motion of the robot with respect to a given degree of freedom; to perform a given mission (e.g., moving the robot from one location to another one), all sub-processes have to synchronize.

With the notable exception of CCS, most process algebras (ACP, CSP, MEIJE, LOTOS...) support multiway synchronization, which is clearly a desirable feature. Yet, these process algebras rely on (associative) binary parallel

composition operators to express concurrency, despite the fact that multiway synchronization between n processes is intrinsically an n-ary operation. The main reason for this situation is probably historical, due to the mathematical origins of process algebras.

For instance, LOTOS has three parallel operators, noted "B_1 |[$\widehat{G}$]| B_2", "B_1 ||| B_2", and "B_1 || B_2", respectively. The first operator is the most general: it expresses that B_1 and B_2 execute concurrently and synchronize only on the gates of $\widehat{G} \cup \{\delta\}$. The second and third operators are particular cases of the former: "|||" corresponds to the case where $\widehat{G}$ is empty (fully asynchronous execution) and "||" to the case where $\widehat{G}$ is the set of all visible gates (fully synchronous execution). From our experience in describing complex, industrial systems using LOTOS, we believe that expressing parallel composition using binary operators has major drawbacks:

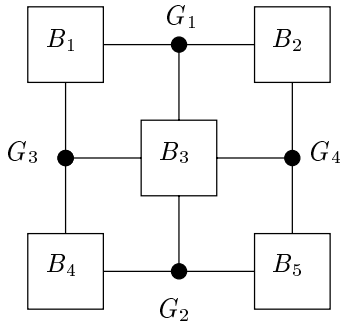

Fig. 1.

– For a given network of concurrent processes, there are usually several different algebraic terms representing this network. For instance, the network shown on Figure 1 can be described using two (equivalent) LOTOS terms, e.g.,

$$((B_1 \ |[G_1]| \ B_2) \ |[G_1, G_3, G_4]| \ B_3) \ |[G_2, G_3, G_4]| \ (B_4 \ |[G_2]| \ B_5)$$

or

$$((B_1 \ |[G_1]| \ B_2) \ ||| \ (B_4 \ |[G_2]| \ B_5)) \ |[G_1, G_2, G_3, G_4]| \ B_3$$

The absence of a canonical form is practically unfortunate, as an algebraic description will strongly depend on the style adopted by its author, thus leading to a lack of uniformity. Moreover, the problem of determining whether

two terms are equivalent is decidable, but not immediate in the general case, as it implies to solve a system of boolean equations [21, 20].

– There are some process networks that can not be expressed as algebraic terms. For instance, the network on Figure 2 can not be expressed using LOTOS parallel composition, because it involves two-by-two synchronization on the same gate G, whereas LOTOS would force all three processes to synchronize on G using a three-way rendez-vous (this is called the *maximal cooperation* paradigm). Sufficient conditions for a process network to be translated into a LOTOS behaviour expression are studied in [2].

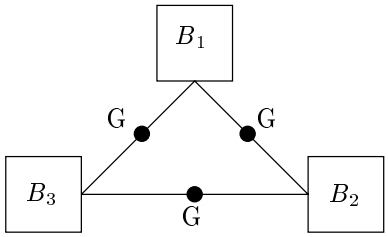

Fig. 2.

Besides these theoretical issues, there are also pragmatic considerations against binary parallel composition operators. The main argument is that binary operators create a discrepancy between the graphical representation of process networks (always present in the designer's mind) and the textual representation as an algebraic term. On the one hand, it is not easy for novice users to write an algebraic term corresponding to a given network of concurrent processes. On the other hand, given an algebraic term, it is not always immediate to infer the corresponding network.

Some network topologies are particularly tricky to express using binary operators. For instance, the simplest algebraic term for representing the token-ring network shown on Figure 3 is:

$$(B_1 \ \texttt{|[}G_1\texttt{]|}\ B_2 \ \texttt{|[}G_2\texttt{]|}\ B_3 \ \texttt{|[}G_3\texttt{]|}\ B_4) \ \texttt{|[}G_4, G_5\texttt{]|}\ B_5$$

which is particularly non-intuitive because the circular symmetry of the network cannot be preserved during the translation to an algebraic term. In this respect, the difficulties inherent to the process algebraic approach have to be compared with graphical formalisms such as SDL [19] or Statecharts [14], in which the user simply has to draw the desired network.

For these reasons, we suggested to introduce in E-LOTOS a new n-ary parallel composition operator that would replace the binary operators of LOTOS. Based on an early suggestion by [3], we made several iterative proposals [10, 31], before

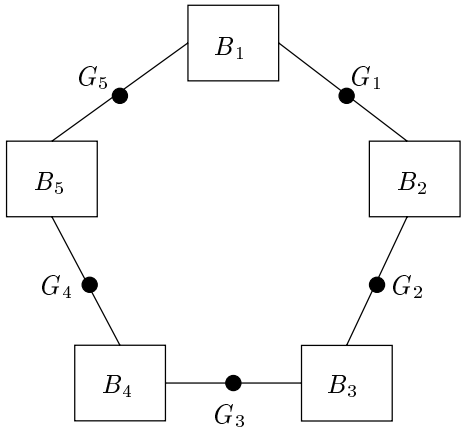

Fig. 3.

our proposal was accepted for being included in E-LOTOS. The basic syntax of our parallel composition operator is:

par
$\quad \widehat{G_1} \rightarrow B_1$
$\quad || \ \widehat{G_2} \rightarrow B_2$
$\quad || \ ...$
$\quad || \ \widehat{G_n} \rightarrow B_n$
endpar

This operator describes a network of $n \geq 1$ concurrent processes $B_1, ..., B_n$. We define I to be the set $\{1, ..., n\}$. To each behaviour B_i is associated an *interface* consisting of a set of gates $\widehat{G_i}$ on which B_i must synchronize. Each $\widehat{G_i}$ can be empty; in such case the arrow before B_i can be omitted.

The operational semantics of this operator is defined by two rules. The first rule expresses that any process B_i can execute asynchronously any action L whose gate G does not belong to the interface $\widehat{G_i}$ and is different from δ (this encompasses the case where $L = \tau$), while the other processes B_j with $j \neq i$ do not evolve:

$$\frac{(\exists L) \ (\exists i \in I) \ B_i \xrightarrow{L} B_i' \ \wedge \ gate(L) \notin \widehat{G_i} \cup \{\delta\} \ \wedge \ (\forall j \in I \backslash \{i\}) \ B_j' = B_j}{\textbf{par } \widehat{G_1} \rightarrow B_1 ... \widehat{G_n} \rightarrow B_n \textbf{ endpar} \xrightarrow{L} \textbf{par } \widehat{G_1} \rightarrow B_1' ... \widehat{G_n} \rightarrow B_n' \textbf{ endpar}}$$

The second rule expresses that a process B_i wanting to execute an action L labelled by a gate $G \in \widehat{G_i} \cup \{\delta\}$ must synchronize with all the other processes B_j such that $G \in \widehat{G_j} \cup \{\delta\}$:

$$\frac{(\exists L) \ (\forall i \in I) \ (\textit{if } gate(L) \in \widehat{G_i} \cup \{\delta\} \textit{ then } B_i \xrightarrow{L} B_i' \textit{ else } B_i' = B_i)}{\textbf{par } \widehat{G_1} \rightarrow B_1 ... \widehat{G_n} \rightarrow B_n \textbf{ endpar} \xrightarrow{L} \textbf{par } \widehat{G_1} \rightarrow B_1' ... \widehat{G_n} \rightarrow B_n' \textbf{ endpar}}$$

This operator solves the aforementioned problems of binary operators by establishing a direct mapping between process networks and their textual representation, thus paving the way for tools that automatically perform the translation from graphical networks to algebraic terms and vice versa. For instance, the networks of Figures 1 and 3 can be expressed as follows:

$$
\begin{array}{l}
\textbf{par} \\
\quad G_1, G_3 \rightarrow B_1 \\
\mid\mid\ G_1, G_4 \rightarrow B_2 \\
\mid\mid\ G_1, G_2, G_3, G_4 \rightarrow B_3 \\
\mid\mid\ G_2, G_3 \rightarrow B_4 \\
\mid\mid\ G_2, G_4 \rightarrow B_5 \\
\textbf{endpar}
\end{array}
$$

and:

$$
\begin{array}{l}
\textbf{par} \\
\quad G_1, G_5 \rightarrow B_1 \\
\mid\mid\ G_1, G_2 \rightarrow B_2 \\
\mid\mid\ G_2, G_3 \rightarrow B_3 \\
\mid\mid\ G_3, G_4 \rightarrow B_4 \\
\mid\mid\ G_4, G_5 \rightarrow B_5 \\
\textbf{endpar}
\end{array}
$$

respectively.

As far as expressiveness is concerned, it is easy to see that the general parallel operator can be obtained as a particular case of our new operator:

$$B_1 \ \mid[\widehat{G}]\mid\ B_2 \ = \ \textbf{par}\ \widehat{G} \rightarrow B_1 \ \mid\mid\ \widehat{G} \rightarrow B_2\ \textbf{endpar}$$

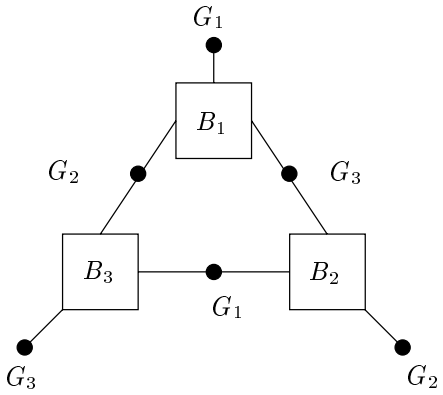

Fig. 4.

Reciprocally, our new operator is strictly more expressive than the "$|\,[\widehat{G}]\,|$" operator of LOTOS. We prove this proposition using the process network shown on Figure 4. This process can easily be described using our new operator:

$$\begin{array}{l}\textbf{par}\\ \quad\quad G_2, G_3 \rightarrow B_1\\ \quad ||\;\; G_1, G3 \rightarrow B_2\\ \quad ||\;\; G1, G2 \rightarrow B_3\\ \textbf{endpar}\end{array}$$

but cannot be expressed using the LOTOS binary parallel operators: to describe this network, one must first synchronize two processes together, then synchronize the result with the third process; assuming that B_1 and B_2 are to be synchronized first (which can be done without loss of generality because the network is symmetric), it is mandatory to synchronize them on gate G_3; then, the resulting term "$B_1 \;|\,[G_3]\,|\; B_2$" has to be synchronized with B_3 on gates G_1 and G_2. But the term obtained does not correspond to the network of Figure 4 where process B_1 can perform actions on gate G_1 independently from process B_3.

As a technical remark, it is worth noticing that the network of Figure 4 could indeed be expressed in LOTOS by defining an auxiliary process P with auxiliary gates G'_1, G'_2, G'_3, and by a clever instantiation of this process so as to rename the auxiliary gates into G_1, G_2, G_3 respectively:

$$\begin{array}{l}P[G_1, G_2, G_3, G_1, G_2, G_3]\\ \textbf{where}\\ \textbf{process}\ P[G_1, G_2, G_3, G'_1, G'_2, G'_3] :=\\ (B_1[G'_1, G_2, G_3] \;\; |\,[G_3]\,|\;\; B_2[G_1, G'_2, G_3]) \;\; |\,[G_1, G_2]\,|\;\; B_3[G_1, G_2, G'_3]\\ \textbf{endproc}\end{array}$$

A similar effect could be achieved using the relabelling operator existing in other process algebras, such as ACP or CSP (in LOTOS, the process instantiation performs relabelling implicitly). However, this solution is probably too tricky for many users; in comparison, our new parallel operator is simpler and more intuitive.

Finally, we slightly extend our operator by allowing to specify a set $\widehat{G_0}$ of synchronization gates which are common to all processes B_i (assuming that $\widehat{G_0} \cap \widehat{G_i} = \emptyset$ for each $i \in I$). This extension is practically helpful for avoiding redundant lists of gates; it is simply defined as a syntactic shorthand:

$$\left(\begin{array}{l}\textbf{par}\ \widehat{G_0}\ \textbf{in}\\ \quad\quad \widehat{G_1} \rightarrow B_1\\ \quad ||\;\; \widehat{G_2} \rightarrow B_2\\ \quad ||\;\; \ldots\\ \quad ||\;\; \widehat{G_n} \rightarrow B_n\\ \textbf{endpar}\end{array}\right) =_{def} \left(\begin{array}{l}\textbf{par}\\ \quad\quad \widehat{G_0} \uplus \widehat{G_1} \rightarrow B_1\\ \quad ||\;\; \widehat{G_0} \uplus \widehat{G_2} \rightarrow B_2\\ \quad ||\;\; \ldots\\ \quad ||\;\; \widehat{G_0} \uplus \widehat{G_n} \rightarrow B_n\\ \textbf{endpar}\end{array}\right)$$

4 From maximal to "m among n" cooperation

Although strictly more expressive than the LOTOS parallel composition operator, our new operator does not allow to represent certain process networks, such as the one of Figure 2. This limitation is unfortunate, because it precludes several networks of practical interest from being modelled, especially the case where a pool of n processes synchronize two by two on the same gate. Although CCS permits such "2 among n" synchronization, other process algebras, such as CSP or LOTOS, do not allow it, because they rely on the maximal cooperation paradigm.

Based on our practical experience, we suggest to extended the parallel composition operator presented in Section 3 in order to allow "m among n" synchronization, i.e., when a set of n processes synchronize m by m on the same gate (with $m \leq n$). Our extended operator is based on our previous proposals [10, 31] submitted to the E-LOTOS standardization committee. This operator has the following syntax:

par $g_1\#m_1, g_2\#m_2, ..., g_p\#m_p$ **in**
  $\widehat{G_1} \rightarrow B_1$
  || $\widehat{G_2} \rightarrow B_2$
  || ...
  || $\widehat{G_n} \rightarrow B_n$
endpar

where $g_1, ..., g_p$ is a (possibly empty) list of gates and where $m_1, ..., m_p$ are natural numbers in the range *1, ..., n* associated to these gates. Each clause "$\#m_j$" is optional: if ommitted, m_j has the default value n. We define $\widehat{G_0}$ to be the gate list $\{g_1, ..., g_p\}$ and we require that $\widehat{G_0} \cap \widehat{G_i} = \emptyset$ for $i \in I$. Notice that we do not require the gates $g_1, ..., g_p$ to be pairwise distinct.

Informally, the semantics of this operator is the following. As regards the gates of $\widehat{G_1} \cup ... \cup \widehat{G_n} \cup \{\delta\}$, this operator behaves exactly as the one described in Section 3. As regards the gates of $\widehat{G_0}$, this operator specifies that processes $B_1, ..., B_n$ can perform m_j among n synchronization on each gate g_j. Two special cases are of interest: if $m_j = 1$, each process B_i can execute asynchronously an action on gate G; if $m_j = n$, all processes B_i have to synchronize on gate G.

To provide a formal semantics, we introduce a predicate noted "$G \rhd J$", where $J \subseteq I$, that is true iff the processes in $\{B_i \mid i \in J\}$ can synchronize together on gate G. Obviously, for a given gate G, there may be several subsets J such that $G \rhd J$. This predicate is defined as follows:

- $\delta \rhd I$, meaning that all concurrent processes must synchronize to terminate, as in LOTOS;
- $(\forall j \in \{1, ..., p\})\ \ (\forall J \subseteq I \mid card(J) = m_j)\ \ g_j \rhd J$, meaning that each gate g_j achieves m_j among n synchronization;
- $(\forall G \in \widehat{G_1} \cup ... \cup \widehat{G_n})\ \ G \rhd \{i \in I \mid G \in \widehat{G_i}\}$, meaning that all processes having G in their interfaces must synchronize on G;

– $(\forall i \in I)\ \ (\forall G \notin \widehat{G_0} \cup \widehat{G_i} \cup \{\delta\})\ \ G \rhd \{i\}$, meaning that each process B_i can perform asynchronously any gate neither mentioned in $\widehat{G_0}$ nor in its interface $\widehat{G_i}$ (δ excepted and τ included).

Using this predicate, the operational semantics of our parallel operator can be defined with a single inference rule:

$$\frac{(\exists L)\ \ (\exists J \subseteq I)\ \ (gate(L) \rhd J)\ \wedge\ ((\forall i \in J)\ \ B_i \xrightarrow{L} B'_i)\ \wedge\ ((\forall i \in I \backslash J)\ \ B'_i = Bi)}{\textbf{par}\ g_j\#m_j... \ \textbf{in}\ \widehat{G_i} \rightarrow B_i... \ \textbf{endpar} \xrightarrow{L} \textbf{par}\ g_j\#m_j... \ \textbf{in}\ \widehat{G_i} \rightarrow B'_i... \ \textbf{endpar}}$$

Using this operator, the process network of Figure 2 can be specified using 2 among 3 synchronization:

par G#2 **in**
  B_1 || B_2 || B_3
endpar

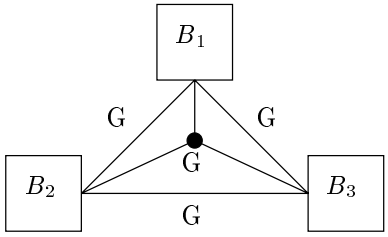

Fig. 5.

More complex process networks, such as the one on Figure 5, in which the same gate G has several degrees of synchronization, can also be described:

par G#2, G#3 **in**
  B_1 || B_2 || B_3
endpar

5 Application

In this Section, we illustrate the application of our parallel operator to the description of the ODP trading function. ODP [18] is a standard framework for distributed applications. Within ISO, E-LOTOS has been developed in the same working group as ODP, and with the intent of being the formal description technique for distributed applications. This explains that ODP-related problems have been a constant source of inspiration for E-LOTOS designers.

Our proposals for introducing "2 among n" synchronization in E-LOTOS [10, 30] was motivated by the highly dynamic nature of ODP systems: processes can be created and destroyed dynamically, and binary communications between processes can be established dynamically. Although it has been argued that such behaviours could only be described by means of mobile process calculi, such as the π-calculus [22, 23], we believe that the most salient aspects of ODP systems can be captured in the framework of a classical process algebra, such as LOTOS, extended with our new parallel operator. A comparative study of both approaches can be found in [9].

The ODP trading function is a typical example of ODP systems: this function is defined informally in an ISO standard [17]. A formal description in E-LOTOS of the most important features of the trading function can be found as an appendix of the E-LOTOS definition document [29, Annex A.3]. In this paper, we focus on the architectural description of the trading function, so as to emphasize how our parallel operator can be used to describe dynamic communication patterns.

The ODP *trader* is a computer process that establishes a relationship between a pool of m *clients* and a pool of n *service providers* (or *servers*) within an open and dynamically changing distributed system.

On the one hand, servers must inform the trader of the services they are ready to offer. Advertising a service offer is called *export*. The trader keeps in a database all the export requests sent by the servers.

On the other hand, clients may ask the trader about available services. Requesting knowledge about a particular service is called *import*. The trader matches the clients' service requests with its database of service offers and, if possible, selects an adequate server. The identification of this server is sent back to the client, which can then contact directly the server without further interaction with the trader.

The interesting issue in this architecture is that the client can eventually communicate with a server the identity of which was unknown to him before asking the trader. In mobile process calculi, this situation can be described using a dynamic creation of mobile gate(s) and/or agent(s). However, alternative approaches are possible, which avoid the complexity of dynamic gate/agent creation. We can model the behaviour of the whole system by the following parallel composition. Let E (export), I (import), and W (work) be three gates used for server-trader, client-trader, and client-server communication respectively. Let "$C[I,W](i)$", "$S[E,W](j)$", and "$T[E,I]$" be three processes representing the i^{th} client, the j^{th} server, and the trader, respectively. The whole architecture can be described by the following term:

```
par E, I in
      T[E, I]
  ||  par W#2 in
            par C[I, W](1)  ||  ...  ||  C[I, W](m) endpar
        ||  par S[E, W](1)  ||  ...  ||  S[E, W](n) endpar
      endpar
endpar
```

It is worth noticing that the two innermost "**par**" operators that define the pool of m clients and the pool of n servers could be expressed in a more concise way using an extended parallel operator that iterates over a finite set of values (such an operator was proposed in [10] and introduced in E-LOTOS). In a more sophisticated modelling, these two "**par**" operators could even be replaced by instantiations of processes creating dynamically a non-bounded number of clients and/or servers (using recursion through parallel composition).

The behaviour of the trader can be described with the following LOTOS process, where *request* and *reply* are two enumerated values indicating the direction of the messages exchanged on gates I and W:

```
process T[E,I](d : DataBase) : noexit :=
  E ?j:Server ?s:Service;
    T[E,I](add_to_database(d,j,s))
  []
  I ?i:Client !request ?s:Service;
    I !i !reply !search_server_in_database(d,s)
      T[E,I](d)
endproc
```

The behaviour of the i^{th} client asking the trader for some service s provided by the j^{th} server, and then requesting this server directly, can be described as follows (we assume that there is always a server available for the requested service):

```
process C[I,W](i : Client) : noexit :=
  I !i !request !s;
    I !i !reply ?j:Server;
      W !j !i !request !s...;
        W !j !i !reply !s...;
          C[I,W](i)
endproc
```

Similarly, the behaviour of the j^{th} server advertising a given service s to the trader and/or answering client requests can be described as follows:

```
process S[E,W](j : Server) : noexit :=
  E !j !s;
    S'[E,W](j)
endproc

process S'[E,W](j : Server) : noexit :=
  W !j ?i:Client !request !s...;
    W !j !i !reply !s...;
      S'[E,W](j)
endproc
```

6 Conclusion

In this study, we have shown that the parallel composition operators found in usual process algebras such as CCS, CSP, ACP, and LOTOS are not so well-suited for an easy description of complex communication patterns. Taking the LOTOS parallel composition operator "$|\,[\widehat{G}]\,|$" as a basis, we suggest to extend this operator in two directions:

- First, we propose to replace the binary operator with an n-ary operator that directly reflects the graphical structure of process networks. From the examples given, it is clear that the n-ary operator is simpler to use by novice users, easier to read (because the structure of process networks is preserved), strictly more expressive, and appropriate for an automatic translation from graphical networks to algebraic terms and vice-versa.
- Second, we increase the expressiveness of this new operator by relaxing the maximal cooperation requirement of CSP and LOTOS, in order to support "m among n" synchronization. Taking the ODP trading function as an example, we show that the new operator is both user-friendly, intuitive, and practically useful.

Our research benefited from discussions in the framework of the E-LOTOS standardization committee. The parallel operator presented in this paper is a refined version of a previous proposal, which we submitted to ISO [10, 31] and which has been integrated in the current version of E-LOTOS [28].

References

1. J. A. Bergstra and J. W. Klop. Process Algebra for Synchronous Communication. *Information and Computation*, 60:109–137, 1984.
2. T. Bolognesi. A Graphical Composition Theorem for Networks of Lotos Processes. In IEEE Computer Society, editor, *Proceedings of the 10th International Conference on Distributed Computing Systems, Washington, USA*, pages 88–95. IEEE, May 1990.
3. Ed Brinksma. *On the Design of Extended LOTOS, a Specification Language for Open Distributed Systems*. PhD thesis, University of Twente, November 1988.
4. J. Camillieri. An Operational Semantics for OCCAM. *International Journal of Parallel Programming*, 18(5):149–167, October 1989.
5. Rance Cleaveland, Eric Madelaine, and Steve Sims. A Front-End Generator for Verification Tools. In Uffe H. Engberg, Kim G. Larsen, and Arne Skou, editors, *Proceedings of TACAS'95 Tools and Algorithms for the Construction and Analysis of Systems (Aarhus, Denmark)*, May 1995. Also available as INRIA Research Report RR-2612.
6. D. Dams and J. F. Groote. Specification and Implementation of Components of a μCRL Toolbox. Technical Report Logic Group Preprint Series 152, Utrecht University, December 1995.
7. Robert de Simone. Higher-level synchronising devices in MEIJE–SCCS. *Theoretical Computer Science*, 37:245–267, 1985.

8. Jean-Claude Fernandez, Hubert Garavel, Alain Kerbrat, Radu Mateescu, Laurent Mounier, and Mihaela Sighireanu. CADP (CÆSAR/ALDEBARAN Development Package): A Protocol Validation and Verification Toolbox. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the 8th Conference on Computer-Aided Verification (New Brunswick, New Jersey, USA)*, volume 1102 of *Lecture Notes in Computer Science*, pages 437–440. Springer Verlag, August 1996.
9. A. Février, E. Najm, G. Leduc, and L. Léonard. Compositional Specification of ODP Binding Objects. In *Proceeding of the 6th IFIP/ICCC Conference on Information Network and Data Communication, INDC'96, Trondheim, Norway*, June 1996.
10. Hubert Garavel. A Wish List for the Behaviour Part of E-LOTOS. Rapport SPECTRE 95-21, VERIMAG, Grenoble, December 1995. Input document [LG5] to the ISO/IEC JTC1/SC21/WG7 Meeting on Enhancements to LOTOS (1.21.20.2.3), Liège (Belgium), December, 18–21, 1995.
11. Hubert Garavel and Mihaela Sighireanu. Towards a Second Generation of Formal Description Techniques – Rationale for the Design of E-LOTOS. In Jan-Friso Groote, Bas Luttik, and Jos van Wamel, editors, *Proceedings of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98 (Amsterdam, The Netherlands)*, pages 187–230, Amsterdam, May 1998. CWI. Invited lecture.
12. J. F. Groote and A. Ponse. Proof theory for μ-CRL. Technical Report CS-9138, CWI Amsterdam, 1991.
13. J. F. Groote and F. W. Vaandrager. Structured Operational Semantics and Bisimulation as a Congruence. *Information and Computation*, 100(2):202–260, October 1992.
14. D. Harel. StateCharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, 1987.
15. C. A. R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.
16. ISO/IEC. LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, September 1988.
17. ISO/IEC. ODP Trading Function. Draft International Standard 13235, ISO — Information Processing Systems, Genève, June 1995.
18. ISO/IEC. Open Distributed Processing — Reference Model. International Standard 10746, ISO — Information Processing Systems, Genève, 1995.
19. ITU-T. Specification and Description Language (SDL). ITU-T Recommendation Z.100, International Telecommunication Union, Genève, 1992.
20. P. Kars. *Process-Algebraic Transformations in Context.* PhD thesis, University of Twente, June 1997.
21. Pim Kars. Representation of Process-Gate Nets in LOTOS and Verification of LOTOS Laws: the Boolean Algebra Approach. In Dieter Hogrefe and Stefan Leue, editors, *Proceedings of the 7th International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols FORTE'94 (Bern, Switzerland)*, October 1994.
22. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes I. *Information and Computation*, 100(1):1–40, September 1992.
23. R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes II. *Information and Computation*, 100(1):41–77, September 1992.
24. Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer Verlag, 1980.

25. Robin Milner. *Communication and Concurrency*. Prentice-Hall, 1989.
26. David Park. Concurrency and Automata on Infinite Sequences. In Peter Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer Verlag, March 1981.
27. G. D. Plotkin. A structural approach to operational semantics. DAIMI FN-19 FN-19, Computer Science Department, Aarhus University, 1981.
28. Juan Quemada, editor. Committee Draft on Enhancements to LOTOS. ISO/IEC JTC1/SC21/WG7 Project 1.21.20.2.3, January 1997.
29. Juan Quemada, editor. Committee Draft on Enhancements to LOTOS. ISO/IEC FCD 15437 (E-LOTOS) (also SC33 N188), April 1998.
30. Mihaela Sighireanu and Hubert Garavel. Defect Report concerning the LOTOS Description of OSI TP Protocol. Rapport SPECTRE 95-18, VERIMAG, Grenoble, December 1995.
31. Mihaela Sighireanu and Hubert Garavel. E-LOTOS User Language. Rapport SPECTRE 96-06, VERIMAG, Grenoble, October 1996. In ISO/IEC JTC1/SC21 Third Working Draft on Enhancements to LOTOS (1.21.20.2.3). Output document of the edition meeting, Kansas City, Missouri, USA, May, 12–21, 1996.
32. C. Vissers, G. Scollo, and M. van Sinderen. Architecture and Specification Style in Formal Descriptions of Distributed Systems. In S. Aggarwal and K. Sabnani, editors, *Proceedings of the 8th International Workshop on Protocol Specification, Testing and Verification (Atlantic City, NJ, USA)*, pages 189–204. IFIP, North-Holland, 1988.