ON GRAPHICAL VS. ALGEBRAIC REPRESENTATIONS OF PROCESS NETWORKS

Technical Report C92-18

April 1992

Tommaso Bolognesi

On graphical vs. algebraic representations of process networks

T. Bolognesi

C.N.R. - CNUCE, Pisa, Italy
(bolog@fdt.cnuce.cnr.it)

Keywords
Distributed systems, formal semantics, process algebras, specification languages.

1. Introduction

In the context of specification languages and process algebras, we consider here two fundamental ways for describing interconnection patterns for sets of interacting and communicating components (processes, processors):

- graph-like box interconnection diagrams, used, for example, in computer-aided software engineering;
- algebraic expressions involving parallel operators, typical of process algebras such as CSP[5] or LOTOS[2, 3].

Diagrams are immediately appealing to the intuition, but often lack a formalized interpretation. Conversely, algebraic behaviour expressions are provided with formal semantics, but, due to their linear nature, are not as flexible as graphs in representing complex interconnection patterns. It is clearly desirable to bridge the gap between the two representations, in order to combine their advantages.

2. Process Interaction Nets (PIN)

We shall take the point of view of process algebras, and understand a *process* as an abstract agent able to perform *actions*, either autonomously or jointly with other processes in its environment. For convenience, we shall use the term action for denoting both single-process actions and multi-process interactions. The behaviour of a process, or of a sets of interacting processes, can be described in terms of *labelled transition systems*, also called *action trees*. These are possibly infinite trees where arcs are labelled by actions.

In order to define process interaction nets, we shall assume the existence of the following sets:

Processes	is the set of *process identifiers*, with typical elements P, Q, R.
Actions	is the set of *action identifiers*, with typical elements a, b, c.

Furthermore, we shall adopt the following notations:

P[A] is a *process instantiation*, where P is a process identifier and A is a list of action identifiers, representing the actions that the process may perform (as done in LOTOS).

$\underline{P} = (P_1[A_1], ..., P_n[A_n])$
is a tuple of process instantiations.

Definition 2.1 A *General Process Interaction Net* (GPIN) over the tuple $\underline{P}$ of process instantiations is an undirected, bipartite graph. Its nodes are partitioned into two sets:

$\underline{P}$ = is the set of *process-nodes*. These are represented by boxes, each one labelled by a different process instantiation.

$\underline{I}$ = is the set of *interaction-nodes*. These are represented by circles, each one labelled by an action identifier. Label duplication is admitted.

Any arc of a GPIN can only connect a process-node with an interaction-node. Some further requirements are necessary, which will become natural in light of the GPIN semantics:

i. Any interaction-node must be connected with at least one process-node.
ii. If two interaction nodes have the same label, they cannot be connected to the same set of process-nodes.
iii. *Positive cooperation condition* : a process-node $P_i[A_i]$ can only be connected to an interaction-node if the action that labels the latter appears in action list A_i. •

Three examples of GPIN are shown in Figure 1.

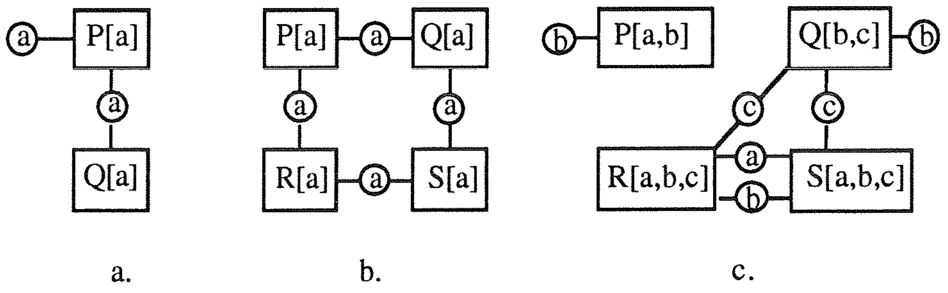

Figure 1 - GPIN's (a, b, c), PIN's (b, c), LOTOS-expressible PIN (c).

If $D \subseteq \{1, ..., n\}$ is a nonempty set of process indices, and *a* is an action, we shall let pair (a, D) denote the star-shaped *multi-arc* rooted at some a-labelled interaction node connected with exactly the D-indexed process nodes.

Behaviour of a GPIN

Let $N_{\underline{P}}$ be a GPIN over the *n* process instantiations in $\underline{P}$. We want now to conceive $N_{\underline{P}}$ itself as a process, and formally define its behaviour by deriving a global transition system from the transition systems of the component processes. This is the usual approach for giving the Structural Operational Semantics (SOS) [8] of process algebraic operators, and, indeed, $N_{\underline{P}}$ can be understood as a powerful algebraic n-ary operator, where process-nodes represent the arguments.

For doing this, we must associate every process instantiation in $\underline{P} = (P_1[A_1], ..., P_n[A_n])$ with a specific transition system. Let $\underline{t} = (t_1, .., t_n)$ be a tuple of labelled transition systems. We say that t_i is *label-consistent* with P_i when the set of labels of t_i is included in A_i. In such case, we can write the *process definition*

$P_i[A_i] := t_i$

(still following the LOTOS convention). By extension, we shall say that $\underline{t}$ is label-consistent with $\underline{P}$, when this is the case element-wise, and write

$\underline{P} := \underline{t}$

for the tuple $(P_1[A_1] := t_1, ..., P_n[A_n] := t_n)$ of process definitions. We shall write $N_{\underline{P}}(\underline{t})$ when we want to refer to a specific definition $\underline{P} := \underline{t}$ of the process-nodes in the net $N_{\underline{P}}$.

As usual, the a-labelled transition by which the labelled transition system t_i turns into some new system t'_i is written: t_i --a--> t'_i . As a further notational convention, we shall write:

$\underline{t}$ --a/D--> $\underline{t}'$,

where $t' = (t_1', ..., t_n')$ is some n-tuple of transition systems also label-consistent with $\underline{P}$, and D is a subset of the process indices $\{1, ..., n\}$, for denoting the set of transitions $\{t_i\text{--a-->}t_i' \mid i \in D\}$, with the implicit assumption that $t_i = t_i'$, for $i \notin D$.

$N_{\underline{P}}(\underline{t})$ shall be able to perform at most the actions that label its interaction-nodes. The idea is that the group of one or more D-indexed processes interconnected by multi-arc (a, D) can collectively perform an a-action whenever each one of these processes is individually able to perform that action. More formally, the inference rule for transition is:

if t--a/D-->t', where (a, D) is a multi-arc of $N_{\underline{P}}$
then $N_{\underline{P}}(\underline{t})$ --a/D--> $N_{\underline{P}}(\underline{t}')$.

Note that, for convenience, we retain also in the second transition label the indication of the set D of *active* process indices. In light of such rule of behaviour, the adequacy of requirements i-iii above is obvious. In particular, requirement iii excludes the existence of interaction-nodes that would in any case be inactive.

Definition 2.2 A *Process-Interaction Net (PIN)* is a GPIN where no two multi-arcs (a, D1), (a, D2) exist such that $D1 \subseteq D2$. •

Such restriction is justified by the results of the next section.

3. Deriving a PIN from a parallel expression

Consistently with the LOTOS approach, we adopt the following syntax for parallel behaviour expressions:

B ::= ProcInst | (B |S| B)

where nonterminal ProcInst represents a process instantiation, as introduced above, |S| is the parallel composition operator, and nonterminal S represents a possibly empty set of synchronization actions on which the two argument expressions (processes) must synchronize.

Example:

```
B    =    ((P[a, b]  |[a]| Q[b, c])
          |[c]|
          (R[a, b, c]  |[a, b]|  (S[a, b, c])
          )
```

We shall sometimes write $B_{\underline{P}}$ for a parallel behaviour expression that includes *one* occurrence of every process instantiation in the tuple $\underline{P}$.

The SOS rules for the LOTOS parallel composition operator are:

(r1)	**if** B1--a-->B1', and $a \notin S$	**then** B1 |S| B2 --a--> B1' |S| B2
(r2)	**if** B2--a-->B2', and $a \notin S$	**then** B1 |S| B2 --a--> B1 |S| B2'
(r3)	**if** B1--a-->B1', B2--a-->B2', and $a \in S$	**then** B1 |S| B2 --a--> B1' |S| B2'

Let $B_{\underline{P}}$ be a parallel expression, and Actions be the set of all action identifiers that occur in its process instantiations or parallel operators. For the example expression above, we have Actions = {a, b, c}. We define now a technique for deriving a GPIN $N_{\underline{P}}$ from $B_{\underline{P}}$.

For each action *a* in Actions, we derive a boolean expression $E(B_{\underline{P}}, a)$ as follows:

- Every process instantiation $P_i[A_i]$ in $B_{\underline{P}}$ is replaced by the simple process identifier P_i, understood as a boolean variable, if *a* is an element of the action set A_i, otherwise it is replaced by symbol '0' (zero).
- Every instance '|S|' of a parallel operator in $B_{\underline{P}}$ is replaced by '*' (understood as a boolean product) if *a* is one of the synchronization actions in S, and by '+' (boolean sum) otherwise.

For our running example, we have:

$$E(B_{\underline{P}}, a) = ((P * 0) + (R * S))$$
$$E(B_{\underline{P}}, b) = ((P + Q) + (R * S))$$
$$E(B_{\underline{P}}, c) = ((0 + Q) * (R + S))$$

Let then $SOP(B_{\underline{P}}, a)$ denote the Sum Of Products obtained by flattening expression $E(B_{\underline{P}}, a)$. For our running example, we have (the '*' symbol is omitted):

$$SOP(B_{\underline{P}}, a) = RS$$
$$SOP(B_{\underline{P}}, b) = P + Q + RS$$
$$SOP(B_{\underline{P}}, c) = QR + QS$$

Furthermore, if D is a non empty index-set $D \subseteq \{1, ..., n\}$, we shall let T_D denote the product term $\pi_{(i \in D)} P_i$ involving the D-indexed process identifiers (their order is immaterial).

The desired net $N_{\underline{P}}$ is obtained as follows. First, create one process-node for each process instantiation in $B_{\underline{P}}$. Then, for every $a \in$ Actions, create a multi-arc (a, D) iff T_D is a product term in $SOP(B_{\underline{P}}, a)$. For our running example, $N_{\underline{P}}$ is shown in Figure 1.c.

The reader may easily check that requirements i-iii of Section 2 are satisfied by the construction above. The importance of PIN's w.r.t. parallel behaviour expressions is captured by the following proposition.

Proposition 3.1 The GPIN $N_{\underline{P}}$ derived from any parallel expression $B_{\underline{P}}$ is a PIN.

Proof By the definition of PIN, we must prove that, for any action *a*, $N_{\underline{P}}$ can not contain two multi-arcs (a, D1), (a, D2) such that D1 $\subseteq$ D2. By the one-to-one correspondence between a-labelled interaction-nodes in $N_{\underline{P}}$ and product-terms in SOP($B_{\underline{P}}$, a), we prove, by induction on the size $|\underline{P}|$ of expression $B_{\underline{P}}$, that no two different (and nonempty) product terms T_{D1} and T_{D2} can be present in SOP($B_{\underline{P}}$, a), such that D1 $\subseteq$ D2.

Basis. Vacuously true: when $B_{\underline{P}}$ consists of a single process instantiation, say $P_1[A_1]$, then Actions = A_1, and SOP($B_{\underline{P}}$, a) is 'P_1' for all a $\in$ Actions.

Step. Let $B_{\underline{P}}$ = (B1 |S| B2), where B1 and B2 are parallel expressions including one or more process instantiations. We have two cases.
If a $\in$ S, then SOP($B_{\underline{P}}$, a) can be derived from expression 'SOP(B1, a) * SOP(B2, a)'. Assuming that T_{D1} and T_{D2} be present in SOP($B_{\underline{P}}$, a), we should have that $T_{D1} = T_{D11} * T_{D12}$, and $T_{D2} = T_{D21} * T_{D22}$, where $T_{D11}, T_{D21} \in$ SOP(B1, a), and $T_{D12}, T_{D22} \in$ SOP(B2, a), and all the terms introduced are nonempty. For the purposes of contradiction, assume that D1 $\subseteq$ D2, that is: D11$\cup$D12 $\subseteq$ D21$\cup$D22. Since the sets of process identifiers appearing in SOP(B1, a) and SOP(B2, a) are disjoint, it must be either D11 $\subseteq$ D21, or D12 $\subseteq$ D22, or both, and the inductive hypothesis is contradicted.
If a $\notin$ S, then SOP($B_{\underline{P}}$, a) can be derived from expression 'SOP(B1, a) + SOP(B2, a)'. If T_{D1} and T_{D2} appear at the same side of the '+' operator, then the inductive hypothesis is contradicted; if they appear at opposite sides, then the property is violated that the sets of process identifiers appearing in SOP(B1, a) and SOP(B2, a) are disjoint. •

4. Equivalence of an expression with its derived PIN

In order to compare at a semantic level parallel composition patterns (be they described by algebraic expressions or interconnection diagrams) in a way which is independent of the actual process definitions, we introduce a new relation. Analogously to the notation $N_{\underline{P}}(\underline{t})$ introduced above, we shall write $B_{\underline{P}}(\underline{t})$ when we want to refer to a specific definition $\underline{P} := \underline{t}$ of the processes in expression $B_{\underline{P}}$.

Definition 4.1 Let $B1_{\underline{P}}$ and $B2_{\underline{P}}$ be two parallel expressions over the tuple $\underline{P}$ of process instantiations. We say that $B1_{\underline{P}}$ and $B2_{\underline{P}}$ are *universally strongly equivalent*, written:

$$B1_{\underline{P}} \sim\sim B2_{\underline{P}}$$

when, for *any* definition $\underline{P} := \underline{t}$ (quantification refers to $\underline{t}$), $B1_{\underline{P}}(\underline{t})$ and $B2_{\underline{P}}(\underline{t})$ are strong bisimulation equivalent [7], written $B1_{\underline{P}}(\underline{t}) \sim B2_{\underline{P}}(\underline{t})$. •

The relation '~~' is an equivalence, trivially, and can also be applied between two PIN's, as well as between a PIN and a behaviour expression, as long as they are defined over the same tuple $\underline{P}$ of process instantiations.

'Only if' part By contradiction. Assume B1~~B2, and N1 ≠ N2. Since the two PIN's have the same process-nodes, they should differ with respect to multi-arcs. We prove that the presence of a *discriminating multi-arc* (a, D) leads to contradiction, by induction on its size |D|.

Basis. Let (a, {k}) be the discriminating multi-arc of size 1, and, for fixing ideas, assume it belong to N1. Consider now the process definitions $\underline{P} := \underline{t}$, where:

t_k	:=	a; stop	
t_i	:=	stop	for $i \neq k$

In such case, we have that N1($\underline{t}$)--a/{k}-->..., due to the presence of the (a, {k}) multi-arc, which enables process P_k to perform action *a* independently of the other processes. However, an a-action is not possible for N2($\underline{t}$): due to the absence of the multi-arc in N2, such an action could only be executed by a group of processes involving at least one process different from P_k, which is impossible, by the definition of the t_i's. Hence, N1 and N2 are not universally strongly equivalent. Thus, the B1~~B2 hypothesis is contradicted.

Step. We show that the existence of a discriminating multi-arc (a, D) of size $m \geq 2$ in N1 (resp. N2) implies the presence of a discriminating multi-arc strictly smaller than *m* in N2 (resp. N1). Consider the process definitions $\underline{P} := \underline{t}$, where:

t_i	:=	a; stop	for $i \in D$
t_i	:=	stop	for $i \notin D$

In such case, we have that N1($\underline{t}$)--a/D-->... By the assumption that N1~~N2, we have that also N2--a/D'-->..., for some index set D', where (a, D') is a multi-arc of N2. Due to the absence of the multi-arc (a, D) in N2, and to the process definitions above, it must be $D \supset D'$. Multi-arc (a, D') cannot exist also in N1, because its co-existence with (a, D) is excluded by Def. 2.2 and Prop. 3.1. Hence, (a, D') is a discriminating multi-arc, and its size is strictly smaller than *m*. •

6. Conclusions

In the context of process algebras, multiple parallel compositions can be conveniently represented by interconnection diagrams, called here (General) Process Interaction Nets, or (G)PIN's, for which a formal semantics is easily provided. While every LOTOS parallel expression B has an associated, universally strong equivalent PIN (Section 3, and Theorem 4.3), not all PIN's admit an equivalent expression. For example, the PIN in Figure 1.b cannot be captured by a LOTOS parallel behaviour expression (the same holds for the GPIN in Figure 1.a). Thus, besides offering the advantages of canonical representations (Theorem 5.1), PIN's (and GPIN's) are more expressive than LOTOS parallel expressions.

A subclass of PIN's, called PGN's (Process Gate Nets) has been studied in [1]. The interaction-nodes of a PGN are called *gates*, and cannot have duplicated labels. Such property is sufficient for *always* guaranteeing the existence of a family of parallel LOTOS expressions equivalent to the net (see [1]). PGN's are currently considered for the standardization of the graphical LOTOS syntax [6].

Graphical representations for process interconnection are studied also in [4]. Process Topology Diagrams (PTD) are more complex than GPIN's, in that their interaction mechanism is based on *two* types of element, called Interaction Points and Communication Indications. Based of such flexibility, PTD's can exhibit a hierarchical structure.

References

[1] T. Bolognesi, "A Graphical Composition Theorem for Networks of LOTOS Processes", *Proceedings of the 10th International Conference on Distributed Computing Systems*, IEEE Computer Society Press, 1990, 88-95.

[2] T. Bolognesi, E. Brinksma, "Introduction to the ISO Specification Language LOTOS", *Computer Networks and ISDN Systems*, Vol. 14, No 1, 1987.

[3] E. Brinksma (ed.), "ISO- Information Processing Systems- Open Systems Interconnection- LOTOS- A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", IS 8807, 1989.

[4] J. Hinterplattner, H. Nirschl, H. Saria, "Process Topology Diagrams", in J. Quemada et al. (eds.), Formal Description Techniques III, North-Holland, 1991, 443-458.

[5] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall, 1985.

[6] E. Najm (ed.), ISO/IEC JTC 1 / SC 21 N. 4871, "G-LOTOS: DAM1 to IS8807 on graphical representations for LOTOS", Jan. 1992.

[7] D. M. R. Park, "Concurrency and Automata on Infinite Sequences", *Lecture Notes on Computer Science*, Vol. 104, Springer, 1981, 167-183.

[8] G. D. Plotkin, "A structural approach to operational semantics", Tech. Rep. DAIMI FN-19, Aarhus Univ., Computer Science Dept., Denmark, 1981.