

Interpreting a Finitary Pi-Calculus in Differential Interaction Nets

Thomas Ehrhard and Olivier Laurent

*Preuves, Programmes & Systèmes
Université Denis Diderot and CNRS*

Abstract

We propose and study a translation of a pi-calculus without sums nor recursion into an untyped version of differential interaction nets. We define a transition system of labeled processes and a transition system of labeled differential interaction nets. We prove that our translation from processes to nets is a bisimulation between these two transition systems. This shows that differential interaction nets are sufficiently expressive for representing concurrency and mobility, as formalized by the pi-calculus.

Our study will concern essentially a replication-free fragment of the pi-calculus, but we shall also give indications on how to deal with a restricted form of replication.

Introduction

Linear Logic proofs [Gir87] admit a *proof net* representation which has a very asynchronous and local reduction procedure, suggesting strong connections with parallel computation. This impression has been enforced by the introduction of *interaction nets* and *interaction combinators* by Lafont in [Laf95].

But the attempts at relating concurrency with linear logic (e.g. [EW97], [AM99], [Mel06], [Bef05], [CF06] based on [FM05]...) missed a crucial feature of concurrency, such as modeled by process calculi like Milner's π -calculus [Mil93], [SW01]: its intrinsic *non-determinism*. Indeed, all known logical systems had either an essentially deterministic reduction procedure – this is the case of intuitionistic and linear logic, and of classical systems such as Girard's LC or Parigot's $\lambda\mu$ – or an excessively non-deterministic one, as Gentzen's classical sequent calculus LK, which equates all proofs of the same formula.

However, many denotational models of the lambda-calculus and of linear logic admit some form of non-determinism (e.g. [Plo76, Gir88b]), showing that a

non-deterministic proof calculus is not necessarily trivial. The first author introduced such models, based on vector spaces (see e.g. [Ehr05]), which have a nice proof-theoretic counterpart, corresponding to a simple extension of the rules that linear logic associates with the exponentials.

In this differential setting, the weakening rule has a mirror image rule called *coweakening*, and similarly for dereliction and for contraction, and the reduction rules have the corresponding mirror symmetry. The corresponding formalism of *differential interaction nets* has been introduced in a joint work by the first author and Regnier [ER06]¹.

In a joint work with Kohei Honda [HL06], the second author proposed a translation of a version of the π -calculus in proof-nets for a version of linear logic extended with the cocontraction rule (as we now understand). The basic idea consists in interpreting the parallel composition as a cut between a contraction link (to which several *outputs* are connected, through dereliction links) and a cocontraction link, to which several promoted receivers are connected. Being promoted, these receivers are replicable, in the sense of the π -calculus. The other fundamental idea of this translation consists in using linear logic polarities for making the difference between outputs (negative) and inputs (positive), and of imposing a strict alternation between these two polarities. This allows to recast in a polarized linear logic setting a typing system for the π -calculus previously introduced by Berger, Honda and Yoshida in [BHY04]. This translation has two features which can be considered as slight defects: it accepts only replicable receivers and it is not really modular (the parallel composition of two processes cannot be described as a combination of the corresponding nets).

Principle of the translation. The purpose of the present paper is to continue this line of ideas, using more systematically the new structures introduced by differential interaction nets².

The first key decision we made, guided by the structure of the typical cocontraction/contraction cut intended to interpret parallel composition, was of

¹ Note that, in this *differential linear logic*, the two additive connectives $\oplus$ and $\&$ are identified, but this does not prevent the system from having good logical properties, and this identification – which results from non-determinism – does not extend to the multiplicative connectives: $\otimes$ and $\wp$ are distinct.

² One should mention here that translations of the π -calculus into nets of various kinds, subject to local reduction relations, have been provided by various authors (cf. the work of Laneve, Parrow and Victor on *solo diagrams* [LPV01], of Beffara and Maurel [BM06], of Milner on *bigraphs* [JM04], of Mazza [Maz05] on *multiport interaction nets* etc.). But these settings have no clear logical grounds nor simple denotational semantics.

associating with each free name of a process not one, but *two* free ports in the corresponding differential interaction net. One of these ports will have a $!$ -type (positive type) and will have to be considered as the *input port* of the corresponding name for this process, and the other one will have a $?$ -type (negative type) and will be considered as an *output port*.

We discovered structures which allow to combine these pairs of wires for interpreting parallel composition and called them *communication areas*: they are obtained by combining in a completely symmetric way cocontraction and contraction cells. There are communication areas of any “arity” (number of pairs of wires connected to it). The communication area of arity 3 can be pictured as in Figure 1, where cocontraction cells are pictured as $!$ -labeled triangles and contraction cells as $?$ -labeled triangles. The ports corresponding to the same pairs are the principal ports of antipodal cells.

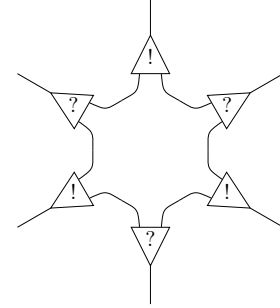


Fig. 1. Communication area

Content. We first introduce differential interaction nets, typed with a recursive typing system (introduced by Danos and Regnier in [Reg92] and which corresponds to the untyped lambda-calculus) for avoiding the appearance of non reducible configurations. For simplifying the presentation, these nets use only a restricted form of the promotion rule of linear logic, which is sufficient for interpreting a replication-free version of the pi-calculus, as well as a restricted form of replication. In this setting, we define a “toolbox”, a collection of nets that we shall combine for interpreting processes, and a few associated reductions, derived from the basic reduction rules of differential interaction nets.

We organize reduction rules of nets as a labeled transition system, whose vertices are nets, and where the transitions correspond to dereliction/codereliction reductions. Then we define a process algebra which is a polyadic π -calculus, without replication and without sums. We specify the operational semantics of this calculus by means of an abstract machine inspired by the machine presented in [AC98, Chapter 16]. We define a transition system whose vertices are the states of this machine, and transitions correspond to input/output reductions. And we define a “translation” relation from machine states to nets and show that this translation relation is a bisimulation between the two transition systems.

Last, we sketch the extension of this translation to a version of our π -calculus augmented with a restricted form of replication (input-guarded replication where the subject is not free in the replicated process, and where all free names of the replicated process are objects of the replication prefix). We conclude

the paper with several concrete examples, showing how various operational features of the π -calculus are modeled in differential interaction nets.

1 Differential interaction nets

We first recall the general syntax of interaction nets, as introduced in [Laf95]. See also [ER06] for more details.

1.1 The general formalism of interaction nets

Assume we are given a set of *symbols* and that an arity (a non-negative integer) and a typing rule is associated with each symbol, this typing rule being a list $(A_0, A_1, \dots, A_n)$ of types (where n is the arity associated to the symbol; types are formulae of some system of linear logic). A *net* is made of *cells*. With each cell γ is associated exactly one symbol and therefore an arity n and a typing rule $(A_0, A_1, \dots, A_n)$. Such a cell γ has one *principal port* p_0 and n *auxiliary ports* $p_1, \dots, p_n$. A net has also a finite set of *free ports*. All these ports (the free ports and the ports associated with cells) have to be pairwise distinct and a set of *wires* is given. This wiring is a family of pairwise disjoint sets of ports of cardinality 2 (ordinary wires) or 0 (loops), and the union of these wires must be equal to the set of all ports of the net. An *oriented wire* of the net is an ordered pair (p_1, p_2) where $\{p_1, p_2\}$ is a wire. In a net, a type is associated with each oriented wire, in such a way that if A is associated with (p_1, p_2) , then $A^\perp$ is associated with (p_2, p_1) . Last, the typing rules of the cells must be respected in the sense that for each cell γ of arity n , whose ports are $p_0, p_1, \dots, p_n$ and typing rule is $(A_0, A_1, \dots, A_n)$, denoting by $p'_0, p'_1, \dots, p'_n$ the ports of the net uniquely defined by the fact that the sets $\{p_i, p'_i\}$ are wires (for $i = 0, 1, \dots, n$), then the oriented wires $(p_0, p'_0), (p'_1, p_1), \dots, (p'_n, p_n)$ have types $A_0, A_1, \dots, A_n$ respectively³.

1.2 Presentation of the cells

Our nets will be typed using a type system which corresponds to the untyped lambda-calculus. This system is based on a single type symbol o (the type

³ This typing discipline admits natural variants: for instance, in typing rules, types could involve type variables, and then satisfying the typing constraints would amount to the existence of a global substitution of type variables by types such that all the required equations hold. But in the present paper, we can restrict to the most basic discipline.

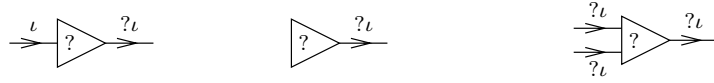
of outputs), subject to the following recursive equation $o = ?o^\perp \wp o$. We set $\iota = o^\perp$, so that $\iota = !o \otimes \iota$ and $o = ?\iota \wp o$.

In the present setting, there are eleven symbols: *par* (arity 2), *bottom* (arity 0), *tensor* (arity 2), *one* (arity 0), *dereliction* (arity 1), *weakening* (arity 0), *contraction* (arity 2), *coderelection* (arity 1), *coweakening* (arity 0), *cocontraction* (arity 2) and *closed promotion* (arity 0). We present now the various cell symbols, with their typing rules, in a pictorial way. The principal port of a cell is located at one of the angles of the triangle representing the cell, the other ports are located on the opposite edge. We put often a black dot to locate the auxiliary port number 1.

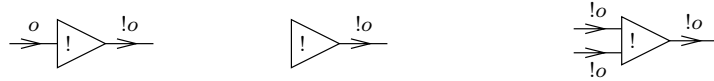
1.2.1 Multiplicative cells. The *par* and *tensor* cells, as well as their “nullary” versions *bottom* and *one* are as follows:



1.2.2 Exponential cells. They are typed according to a strictly polarized discipline. Here are first the *why not* cells, which are called *dereliction*, *weakening* and *contraction*:



and then the *bang* cells, called *coderelection*, *coweakening* and *cocontraction*:



1.2.3 Closed promotion cells and the definition of nets. The notion of simple net is then defined inductively, together with the notion of *closed promotion* cell.

Given a simple⁴ net s with only one free port $\boxed{s} \multimap^o$ we introduce a cell $\boxed{s!} \multimap^{!o}$. This corresponds to the *promotion box* construction of linear logic nets, it is restricted here to the case where the resulting box has no “auxiliary ports”. We say that s is the subnet of this promotion cell. There would be of course no difficulties in introducing more general promotion cells, with auxiliary ports, but we shall not use them in the present work.

⁴ In general, one should also admit non simple nets here, but this is not necessary for the purpose of the present work.

A *simple net* is a typed interaction net, in the signature we have just defined.

A *net* is a finite formal sum of simple nets having all the same interface. Remember that the interface of a simple net s is the set of its free ports, together with the mapping associating to each free port the type of the oriented wire of s whose ending point is the corresponding port.

1.2.4 Labeled nets. Let $\mathcal{L}$ be a countable set of labels containing a distinguished element τ (to be understood as the absence of label). A *labeled simple net* is a simple net where all dereliction, codereliction and promotion cells are equipped with labels belonging to $\mathcal{L}$.

We say that a simple net s satisfies the *condition on labels for simple nets* if two labels associated with distinct cells⁵ of s are either distinct or equal to τ . As such, this condition is not preserved under reduction, due to the fact that promotion cells are duplicated. Therefore, we reinforce this condition by requiring also that all the promotion cells of s be labeled by τ and all the labels occurring in subnets of promotion cells of s be equal to τ . We shall refer to the conjunction of these conditions as to the CLB (*condition on labels and boxes*).

All the nets we consider in this paper are labeled. In our pictures, the labels of dereliction, codereliction and box cells will be indicated, unless it is τ , in which case the (co)dereliction or box cell will be drawn without any label.

2 Reduction rules

We denote by Δ the collection of all simple nets, ranged over by the letters s, t, u , with or without subscripts or superscripts, and by $\mathbb{N}\langle\Delta\rangle$ the collection of all nets (finite sums of simple nets with the same interface), ranged over by the letters S, T, U , with or without subscripts or superscripts. We consider Δ as a subset of $\mathbb{N}\langle\Delta\rangle$ ($s \in \Delta$ being identified with the sum made of exactly one copy of s).

A *reduction rule* is a subset $\mathcal{R}$ of $\Delta \times \mathbb{N}\langle\Delta\rangle$ consisting of pairs (s, S) where s is a simple net made of two cells connected by their principal ports and S is a net that has the same interface as s . This set $\mathcal{R}$ can be finite or infinite. Such a relation is easily extended to arbitrary simple nets ($s \mathcal{R} T$ if there is $(s_0, u_1 + \dots + u_n) \in \mathcal{R}$ where s_0 is a subnet of s , each u_i is a simple net and $T = t_1 + \dots + t_n$ where t_i is the simple net obtained by replacing s_0 by u_i in

⁵ This mean that they can also occur in subnets associated with promotion cells, at any depth.

s). This relation is extended to nets (sums of simple nets): $s_1 + \dots + s_n$ (where each s_i is simple) is related to T by this extension $\mathcal{R}^\Sigma$ if $T = T_1 + \dots + T_n$ where, for each i , $s_i \mathcal{R} T_i$ or $s_i = T_i$. Last, $\mathcal{R}^*$ is the transitive closure of $\mathcal{R}^\Sigma$ (which is reflexive), and we use $\mathcal{R}^+$ for $\mathcal{R}^* \setminus \text{Id}$ (where Id is the identity relation).

2.1 Defining the reduction

2.1.1 Multiplicative reduction. The first two rules concern the interaction of two multiplicative cells of the same arity.

$$\begin{array}{c} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \sim_m \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \\ \begin{array}{c} \perp \\ \perp \\ \perp \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \perp \\ \perp \\ \perp \end{array} \xrightarrow{o} \begin{array}{c} \perp \\ \perp \\ \perp \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \perp \\ \perp \\ \perp \end{array} \xrightarrow{o} \begin{array}{c} \perp \\ \perp \\ \perp \end{array} \sim_m \varepsilon \end{array}$$

where ε stands for the empty simple net (not to be confused with the net $0 \in \mathbb{N}\langle\Delta\rangle$, the empty sum, which is not a simple net). The next two rules concern the interaction between a binary and a nullary multiplicative cell.

$$\begin{array}{c} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \sim_m \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \\ \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \sim_m \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \end{array}$$

So here the reduction rule (denoted as $\sim_m$) has four elements.

2.1.2 Communication reduction. Let $R \subseteq \mathcal{L}$. We have the following reductions if $l, m \in R$.

$$\begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \sim_{c,R} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array}$$

So the set $\sim_{c,R}$ is in bijective correspondence with R^2 .

2.1.3 Non-deterministic reduction. Let $R \subseteq \mathcal{L}$. We have the following reductions if $l \in R$.

$$\begin{array}{c} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \sim_{nd,R} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} + \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \xrightarrow{o} \begin{array}{c} \text{?}\iota \\ \text{?}\iota \\ \text{?}\iota \end{array} \\ \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \sim_{nd,R} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} + \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagup \\ \diagdown \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \begin{array}{c} \diagdown \\ \diagup \end{array} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \xrightarrow{o} \begin{array}{c} \text{!}o \\ \text{!}o \\ \text{!}o \end{array} \end{array}$$

$$\begin{array}{ccc} \begin{array}{c} \xrightarrow{l} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} & \xrightarrow{\text{nd}, R} 0 & \begin{array}{c} \xrightarrow{o} \triangleleft \text{!} \xrightarrow{!o} \triangleleft \text{?} \end{array} \xrightarrow{\text{nd}, R} 0 \end{array}$$

Remark 1 These rules are particularly interesting. If one considers a sum $s_1 + \dots + s_n$ of several simple nets as a non-deterministic superposition, then a reduction $s \rightsquigarrow s_1 + \dots + s_n$ can simply mean that all the reductions $s \rightsquigarrow s_1, \dots, s \rightsquigarrow s_n$ are possible. This is certainly a reasonable interpretation of the sum. However, this interpretation hides a crucial point: a simple net s can contain a lot of redexes. By choosing one of these redexes various reductions are therefore possible, starting from s : one can have e.g. $s \rightsquigarrow T_1, \dots, s \rightsquigarrow T_p$ (here, the T_j s can be simple or non simple nets). But *this does not mean that* $s \rightsquigarrow T_1 + \dots + T_p$.

This crucial distinction between two forms of non-determinism in computation is one of the most original features of this new setting. Such an explicit dichotomy is typically absent from the process algebra viewpoint on concurrency. It seems that the same kind of distinction is implemented by the notion of *conflict* in event structures [Win87].

2.1.4 Structural reduction.

$$\begin{array}{ccc} \begin{array}{c} \xrightarrow{?l} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} & \xrightarrow{\sim_s} & \begin{array}{c} \xrightarrow{?l} \triangleleft \text{!} \\ \xrightarrow{?l} \triangleleft \text{!} \end{array} \quad \begin{array}{c} \xrightarrow{!o} \triangleleft \text{!} \xrightarrow{!o} \triangleleft \text{?} \end{array} \xrightarrow{\sim_s} \begin{array}{c} \xrightarrow{!o} \triangleleft \text{?} \\ \xrightarrow{!o} \triangleleft \text{?} \end{array} \\ \\ \begin{array}{c} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} \xrightarrow{\sim_s} \varepsilon \quad \begin{array}{c} \xrightarrow{?l} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} \xrightarrow{\sim_s} \begin{array}{c} \xrightarrow{?l} \triangleleft \text{!} \text{ } s! \\ \xrightarrow{?l} \triangleleft \text{!} \text{ } s! \end{array} \\ \\ \begin{array}{c} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} \xrightarrow{\sim_s} \varepsilon \quad \begin{array}{c} \xrightarrow{?l} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} \xrightarrow{\sim_s} \begin{array}{c} \xrightarrow{?l} \triangleleft \text{!} \xrightarrow{?l} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \\ \xrightarrow{?l} \triangleleft \text{!} \xrightarrow{?l} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} \end{array}$$

We use $\sim_s$ for the symmetric and transitive closure of $\rightsquigarrow_s$.

2.1.5 Box reduction. Let $R \subseteq \mathcal{L}$. We have the following reductions if $l, m \in R$.

$$\begin{array}{c} \xrightarrow{l} \triangleleft \text{?} \xrightarrow{?l} \triangleleft \text{!} \end{array} \xrightarrow{\text{b}, R} \begin{array}{c} \xrightarrow{l} \text{ } s \end{array}$$

Observe that the reduction rules are compatible with the identification of the coweakening cell with a promotion cell containing the 0 net. Observe also that

the only rules which do not admit a “symmetric” rule are those which involve a promotion cell. Indeed, promotion is the only asymmetric rule of differential linear logic.

2.1.6 Preservation of the CLB. One can check that we have provided reduction rules for all redexes compatible with our typing system: for any simple net s made of two cells connected through their principal ports, there is a reduction rule whose left member is s . This rule is unique, up to the choice of a set of labels, but this choice has no influence on the right member of the rule.

One can also check, by simple inspection of the rules that, if t is a simple net which satisfies the CLB (see 1.2.4) and if $t \rightsquigarrow t_1 + \dots + t_n$ by one of our reduction rules, then all the simple nets t_i satisfy the CLB.

2.2 Confluence

Theorem 2 *Let $R, R' \subseteq \mathcal{L}$. Let $\mathcal{R} \subseteq \Delta \times \mathbb{N}\langle\Delta\rangle$ be the union of some of the reduction relations $\rightsquigarrow_{c,R}$, $\rightsquigarrow_{nd,R'}$, $\rightsquigarrow_m$, $\rightsquigarrow_s$ and $\rightsquigarrow_{b,R}$. The relation $\mathcal{R}^*$ is confluent on $\mathbb{N}\langle\Delta\rangle$.*

The proof is essentially trivial since the rewriting relation has no critical pair (see [ER06]). Given $R \subseteq \mathcal{L}$, we consider in particular the following reduction: $\rightsquigarrow_R = \rightsquigarrow_m \cup \rightsquigarrow_{c,\{\tau\}} \cup \rightsquigarrow_s \cup \rightsquigarrow_{b,\{\tau\}} \cup \rightsquigarrow_{nd,R}$. We set $\rightsquigarrow_d = \rightsquigarrow_\emptyset$ (“d” for “deterministic”) and denote by $\sim_d$ the symmetric and transitive closure of this relation.

Some of the reduction rules we have defined depend on a set of labels. This dependence is clearly monotone in the sense that the relation becomes larger when the set of labels increases.

2.3 A transition system of simple nets

2.3.1 Restriction on simple nets. From now on, and until Section 6, we assume that all simple nets satisfy the CLB; remember that, together, these conditions are preserved under reduction. This will be sufficient for dealing with replication-free processes. The reason for this restriction is that the useful Lemmata 3 and 4 seem to depend on the uniqueness of label occurrences.

2.3.2 $\{l, m\}$ -neutrality. Let l and m be distinct elements of $\mathcal{L} \setminus \{\tau\}$. We call (l, m) -communication redex a communication redex whose coderelection cell is labeled by l and whose dereliction cell is labeled by m .

The following is a simple, but quite useful remark.

Lemma 3 *Let s_0 be a simple net which contains an (l, m) -communication redex. If $s_0 \rightsquigarrow_{\{l, m\}}^* T_0$, then T_0 is a simple net t_0 which contains an (l, m) -communication redex and one has actually $s_0 \rightsquigarrow_d^* t_0$. Moreover, if s is the simple net obtained from s_0 by reducing the (l, m) -communication redex, then $s \rightsquigarrow_d t$ where t is the simple net obtained from t_0 by reducing the (l, m) -communication redex.*

We say that a simple net s is $\{l, m\}$ -neutral if, whenever $s \rightsquigarrow_{\{l, m\}}^* S$, none of the simple summands of S contains an (l, m) -communication redex.

Lemma 4 *Let s be a simple net. If $s \rightsquigarrow_{\{l, m\}}^* S$ where all the simple summands of S are $\{l, m\}$ -neutral, then s is also $\{l, m\}$ -neutral.*

Proof. Assume, towards a contradiction, that $s \rightsquigarrow_{\{l, m\}}^* T = s_1 + \dots + s_n$ where each s_i is simple and where s_1 contains an (l, m) -communication redex. By the Church-Rosser property of $\rightsquigarrow_{\{l, m\}}^*$, there is S' such that $T \rightsquigarrow_{\{l, m\}}^* S'$ and $S \rightsquigarrow_{\{l, m\}}^* S'$. By Lemma 3 applied to s_1 , S' must have a summand containing an (l, m) -communication redex, contradicting our hypothesis on S . $\square$

2.3.3 The transition system. We define a labeled transition system $\mathbb{D}_{\mathcal{L}}$ whose objects are simple nets, and transitions are labeled by pairs of distinct elements of $\mathcal{L} \setminus \{\tau\}$. Let s and t be simple nets, we have $s \xrightarrow{\overline{lm}} t$ if the following holds: $s \rightsquigarrow_{\{l, m\}}^* s_1 + s_2 + \dots + s_n$ where s_1 is a simple net which contains an (l, m) -communication redex and becomes t when one reduces this redex, and each s_i (for $i > 1$) is $\{l, m\}$ -neutral.

Lemma 5 *The relation $\sim_d \subseteq \Delta \times \Delta$ is a strong bisimulation on $\mathbb{D}_{\mathcal{L}}$.*

Proof. Let $s, s' \in \Delta$ and assume that $s \sim_d s'$. Assume moreover that $s \xrightarrow{\overline{lm}} t$, which means that $s \rightsquigarrow_{\{l, m\}}^* s_0 + s_1 + \dots + s_n$ where each s_i is simple, s_0 contains an (l, m) -communication redex, each s_i is $\{l, m\}$ -neutral for $i \geq 1$ and t is obtained by reducing the (l, m) -communication redex of t_0 . By the Church-Rosser property of $\rightsquigarrow_{\{l, m\}}^*$ (remember that $\sim_d \subseteq \rightsquigarrow_{\{l, m\}}^*$), there exists $U \in \mathbb{N}\langle \Delta \rangle$ such that $s_0 + s_1 + \dots + s_n \rightsquigarrow_{\{l, m\}}^* U$ and $s' \rightsquigarrow_{\{l, m\}}^* U$. But by Lemmata 3 and 4, we have $U = u_0 + u_1 + \dots + u_m$ with $s_0 \rightsquigarrow_d u_0$, u_0 contains an (l, m) -communication redex, and if we reduce this redex, we obtain a net t' such that $t \rightsquigarrow_d t'$. $\square$

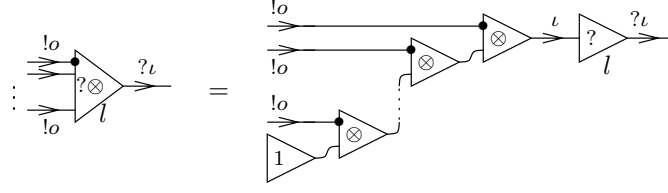


Fig. 2. Dereliction-tensor compound cell

3 A toolbox for process calculi interpretation

We introduce now a few families of simple nets, which are built using the previously introduced basic cells. They will be used as basic modules for interpreting processes. All of these nets, but the communication areas, can be considered as *compound cells*: in reduction, they behave in the same way as cells of interaction nets.

3.1 Compound cells

3.1.1 Generalized contraction and cocontraction. A *generalized contraction cell* or *contraction tree* is a simple net γ (with one principal port and a finite number of auxiliary ports) which is either a wire or a weakening cell or a contraction cell whose auxiliary ports are connected to the principal port of other contraction trees, whose auxiliary ports become the auxiliary ports of γ . Generalized cocontraction cells (cocontraction trees) are defined dually.

We use the same graphical notations for generalized (co)contraction cells as for ordinary (co)contraction cells, with a “*” in superscript to the “!” or “?” symbols to avoid confusions. Observe that there are infinitely many generalized (co)contraction cells of any given arity.

3.1.2 The dereliction-tensor and the codereliction-par cells. Let n be a non-negative integer. We define an n -ary $?\otimes$ compound cell as in Figure 2. It will be decorated by the label of its dereliction cell (if different from τ). The number of tensor cells in this compound cell is equal to n . One defines dually the $!\wp$ compound cell.

3.1.3 The prefix cells. Now we can define the compound cells which will play the main role in the interpretation of prefixes of the π -calculus. Thanks to the above defined cells, all the oriented wires of the nets we shall define will bear type $?\iota$ or $!o$. Therefore, we adopt the following graphical convention: the wires will bear an orientation corresponding to the $?\iota$ type.

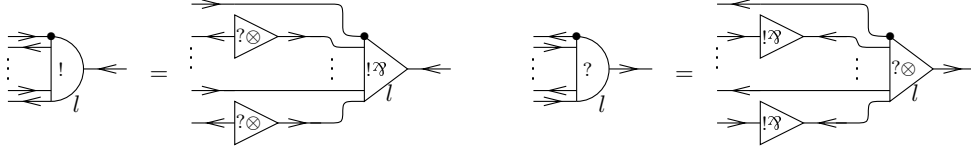


Fig. 3. Input and output compound cells

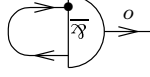


Fig. 4. Identity

The n -ary input cell and the n -ary output cell are defined in Figure 3, they have n pairs of auxiliary ports. In Section 6, we shall also use a version of the input prefix where the coderelection cell has been removed. The main port of this *pre-input cell* bears therefore the type o (when oriented towards the outside) instead of the type $!o$. We use the same notation as for the input cell (Figure 3), with the only difference that the symbol “!” will be replaced by the symbol “ $\overline{}$ ”. See an example in Figure 4.

Prefix cells are labeled by the label carried by their outermost $?⊗$ or $!⧻$ compound cell, if different from τ , the other $?⊗$ or $!⧻$ compound cells being unlabeled (that is, labeled by τ).

3.1.4 Transistors and boxed identity. In order to implement the sequentiality corresponding to sequences of prefixes in the π -calculus, we shall use the unary output prefix cell defined above as a kind of transistor, that is, as a kind of switch that one can put on a wire, and which is controlled by another wire. This idea is strongly inspired by the translation of the π -calculus in the calculus of solos⁶.

These switches will be closed by “boxed identity cells”, which are the unique use we make of promotion in the present work (apart from the extension sketched in Section 6). Let I be the “identity” net of Figure 4, which uses a pre-input compound cell. Then we shall use the closed promotion cell labeled by $I^!$: $I^! : \boxed{I} \rightarrow$.

3.2 Communication tools

⁶ It is shown in [LV03] that one can encode the π -calculus sequentiality induced by prefix nesting in the completely asynchronous solo formalism: the idea of such translations is to observe that, in a solo process like $P = \nu y(u(x, y) \mid y(\dots)) \mid Q$, the first solo must interact before the second one with the environment Q .

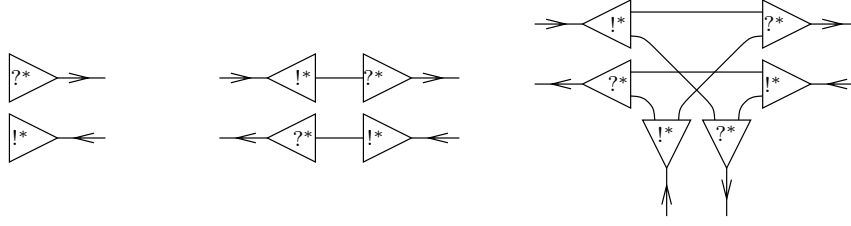


Fig. 6. Communication areas of order -1 , 0 and 1

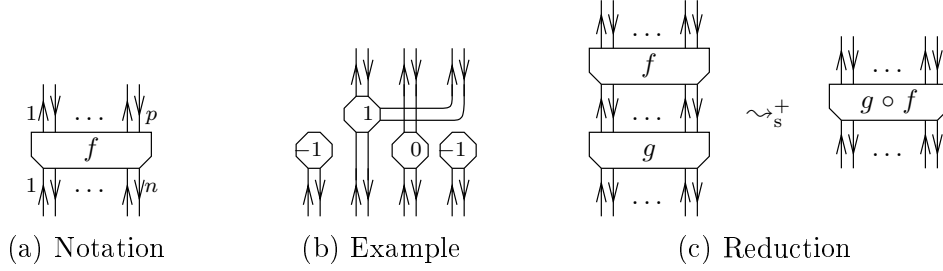


Fig. 7. Identification structures

3.2.1 The communication areas. Let $n \geq -2$. We define a family of nets with $2(n + 2)$ free ports, called *communication areas of order n* , that we shall draw using rectangles with beveled angles. Figure 5 shows how we picture a communication area of order 3.

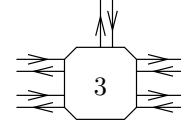


Fig. 5. Area of order 3

A communication area of order n is made of $n + 2$ pairs of $(n + 1)$ -ary generalized cocontraction and contraction cells $(\gamma_1^+, \gamma_1^-), \dots, (\gamma_{n+1}^+, \gamma_{n+1}^-)$, with, for each i and j such that $1 \leq i < j \leq n + 2$, a wire from an auxiliary port of γ_i^- to an auxiliary port of γ_j^- and a wire from an auxiliary port of γ_i^+ to an auxiliary port of γ_j^+ .

So the communication area of order -2 is the empty net ε , and communication areas of order -1 , 0 and 1 are the structures given in Figure 6.

3.2.2 Identification structures. Let $n, p \in \mathbb{N}$ and let $f : \{1, \dots, p\} \rightarrow \{1, \dots, n\}$ be a function. An *f-identification net* is a structure with $p + n$ pairs of free ports (p pairs correspond to the domain of f and, in our pictures, will be attached to the non beveled side of the identification structure, and n pairs correspond to the codomain of f , attached to the beveled side of the structure) as in Figure 7(a). Such a net is made of n communication areas, and on the j 'th area, the j 'th pair of wires of the codomain is connected, as well as the pairs of wires of index i of the domain such that $f(i) = j$. For instance, if $n = 4$, $p = 3$, $f(1) = 2$, $f(2) = 3$ and $f(3) = 2$, a corresponding identification structure is made of four communication areas, two of order -1 , one of order 0 and one of order 1 , as in Figure 7(b).

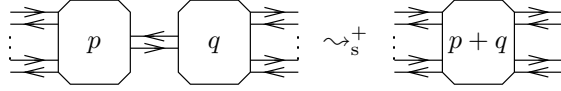


Fig. 8. Aggregation, with $p, q \geq -1$

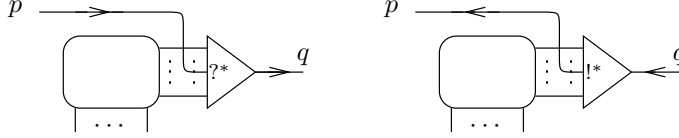


Fig. 9. Port forwarding

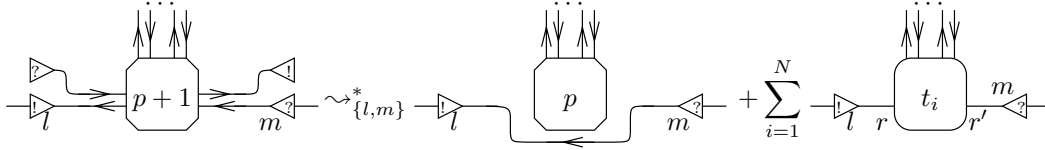


Fig. 10. Dereliction and codereliction communicating through a communication area

3.3 Useful reductions.

3.3.1 Aggregation of communication areas. One of the nice properties of communication areas is that, when one connects two such areas through a pair of wires, one gets another communication area; if the two areas are of respective orders $p \geq -1$ and $q \geq -1$, the resulting area is of order $p + q$, see Figure 8.

3.3.2 Composition of identification structures. In particular, we get the reduction of Figure 7(c).

3.3.3 Port forwarding in a net. Let t be a net and p be a free port of t . We say that p is *forwarded in t* if there is a free port q of t such that t is of one of the two shapes given in Figure 9.

3.3.4 Forwarding of derelictions and coderelictions in communication areas. The reduction of Figure 10 shows that derelictions and coderelictions can meet each other, when connected to a common communication area. More precisely, let $l, m \in \mathcal{L}$, then we have the reduction of Figure 10, where N is a non-negative integer (actually, $N = (p+1)^2$) and, in each simple net t_i , both ports r and r' are forwarded.

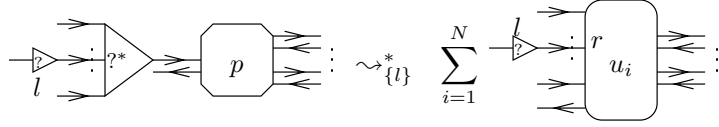


Fig. 11. General forwarding

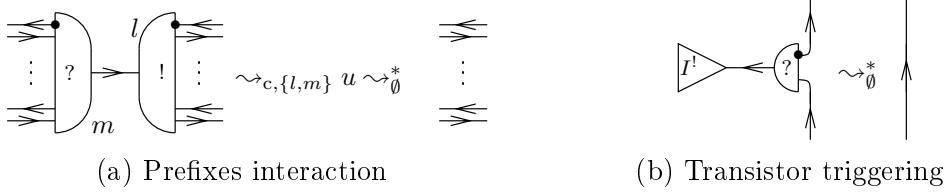


Fig. 12. Prefix reduction

3.3.5 General forwarding. Let $l \in \mathcal{L}$. The more general but less informative property shown in Figure 11 will also be used, where in each simple net u_i , the port r is forwarded (see 3.3.3). Of course one also has a dual reduction (where the dereliction is replaced by a codereliction, and the generalized contraction by a generalized cocontraction).

3.3.6 Reduction of prefixes. Let $l, m \in \mathcal{L}$. If we connect an n -ary output prefix labeled by m to a p -ary input prefix labeled by l , we obtain a net which reduces by $\sim_{c, \{l, m\}}$ to a net u which reduces by $\sim_{\{\tau\}}^*$ to 0 if $n \neq p$ and to simple wires, in Figure 12(a), if $n = p$.

3.3.7 Transistor triggering. A boxed identity connected to the principal port of a unary output cell used as a “transistor” turns it into a simple wire as in Figure 12(b).

4 A polyadic finitary π -calculus and its encoding

The process calculus we consider is a fragment of the π -calculus where we have suppressed the following features: sums, replication, recursive definitions, match and mismatch. This does not mean of course that differential interaction nets cannot interpret these features⁷. Let $\mathcal{N}$ be a countable set of names. Our processes are defined by the following syntax. We use the same set $\mathcal{L}$ of labels as before.

- nil is the empty process.

⁷ A restricted form of replication can be interpreted using exponential boxes as we shall see in Section 6, sums are probably related to the unique additive connective of differential linear logic.

- If P_1 and P_2 are processes, then $P_1 \mid P_2$ is a process.
- If P is a process and $a \in \mathcal{N}$, then $\nu a \cdot P$ is a process. The name a is bound in this process.
- If P is a process, $a, b_1, \dots, b_n \in \mathcal{N}$, the b_i s being pairwise distinct and if $l \in \mathcal{L}$, then $Q = [l]a(b_1 \dots b_n) \cdot P$ is a process (prefixed by an input action, whose subject is a and whose objects are the b_i s; the name a is free and each b_i is bound in Q and hence a is distinct from each b_i).
- If P is a process, $a, b_1, \dots, b_n \in \mathcal{N}$ and $l \in \mathcal{L}$, then $\overline{[l]}a\langle b_1 \dots b_n \rangle \cdot P$ is a process (prefixed by an output action, whose subject is a and whose objects are the b_i s). This construction does not bind the names b_i , and one does not require the b_i s to be distinct. The name a can be equal to some of the b_i s.

We introduce this labeling of prefixes to distinguish the various occurrences of names as subject of prefixes; these labels do not play any active role in the reduction of processes, they are here only for tracing purposes. The set $\text{FV}(P)$ of free names of a process P is defined in the obvious way. The α -equivalence relation on processes is defined as usual.

A *labeled process* is a process where all prefixes are labeled, by pairwise distinct labels, all these labels being different from τ . If P is a labeled process, $\mathcal{L}(P)$ denotes the set of all labels occurring in P . Observe that this set has a natural poset (forest actually) structure ($l < m$ if, in P , l labels a prefix μ and m occurs in the process prefixed by μ).

All the processes we consider in this paper are labeled.

4.1 Arity typing of processes.

Although not strictly necessary, it is convenient to assume that our processes are “typed” in the sense that each name is given with an arity, which is a possibly empty list of arities. When a name of arity $(\rho_1, \dots, \rho_n)$ occurs as subject, it is always assumed that it has n objects $b_1, \dots, b_n$, the arity of b_i being ρ_i . This guarantees that, during the reduction, when an input prefix communicates with an output prefix, the numbers of objects of the two involved prefixes coincide. Since this is a standard π -calculus notion (see [SW01], Part III), we shall not say more about it, and we shall simply assume that, during the reduction of processes and states, the arities of communicating prefixes always coincide.

4.2 An execution model

Rather than considering a rewriting relation on processes as one usually does, we prefer to define an “environment machine”, similar to the machine introduced in [AC98, Chapter 16]⁸.

An *environment* is a function e from a finite subset $\text{Dom } e$ of $\mathcal{N}$ to a finite subset $\text{Codom } e$ of $\mathcal{N}$. A *closure* is a pair (P, e) where P is a process and e is an environment such that $\text{FV}(P) \subseteq \text{Dom}(e)$. A *soup* is a multiset $\Gamma = (P_1, e_1) \cdots (P_N, e_N)$ of closures (denoted by simple juxtaposition). The set $\text{FV}(\Gamma)$ of free names of a soup Γ is the union of the codomains of the environments of Γ . The soup Γ is labeled if all the P_i s are labeled, with pairwise disjoint sets of labels. A *state* is a pair (Γ, L) where Γ is a soup and L is a set of names (the names which have to be considered as local to the state) and we set $\text{FV}(\Gamma, L) = \text{FV}(\Gamma) \setminus L$. The state (Γ, L) is labeled if the soup Γ is labeled.

All the states we consider are labeled. One defines the poset $\mathcal{L}(\Gamma, L)$ of all labels of the state (Γ, L) in the straightforward way, as the parallel composition of the posets associated with the processes of the closures of Γ .

4.2.1 α -equivalence of states. Given a partial function $f : \mathcal{N} \rightarrow \mathcal{N}$ and a process P , we denote by $f \cdot P$ the process where each free name a has been replaced by $f(a)$ (if $a \in \text{Dom } f$) — this construction is not part of the syntax, it is a meta-operation like substitution in the lambda-calculus —. Of course, bound names have to be renamed to avoid name clashes.

Two closures (P_1, e_1) and (P_2, e_2) are α -equivalent (written $(P_1, e_1) \sim_\alpha (P_2, e_2)$) if there is a bijection on names f such that $f \cdot P_1$ and P_2 are α -equivalent, and $e_2 \circ f = e_1$. Two soups Γ and Δ are α -equivalent if $\Gamma = \gamma_1 \dots \gamma_N$ and $\Delta = \delta_1 \dots \delta_N$ with $\gamma_i \sim_\alpha \delta_i$ for each i . Let $f : \mathcal{N} \rightarrow \mathcal{N}$ be a function. If $\gamma = (P, e)$ is a closure, one sets $f \cdot \gamma = (P, f \circ e)$. And last, $f \cdot (\gamma_1 \dots \gamma_N) = (f \cdot \gamma_1) \cdots (f \cdot \gamma_N)$.

Two states (Γ, L) and (Δ, M) are α -equivalent if there is a bijection on names f which is the identity on $\mathcal{N} \setminus L$ and satisfies $f(L) = M$ and $f \cdot \Gamma \sim_\alpha \Delta$.

4.2.2 Canonical form of a state. We say that a process is *guarded* if it starts with an input prefix or an output prefix. We say that a soup $\Gamma = (P_1, e_1) \cdots (P_N, e_N)$ is *canonical* if each P_i is guarded, and that a state (Γ, L)

⁸ The reason for this choice is that the rewriting approach uses an operation which consists in replacing a name by another name in a process. The corresponding operation on nets is rather complicated and we prefer not to define it here.

is canonical if the soup Γ is canonical. One defines a rewriting relation $\leadsto_{\text{can}}$ which allows to turn a state into a canonical one.

$$\begin{aligned} & ((\text{nil}, e)\Gamma, L) \leadsto_{\text{can}} (\Gamma, L) \\ & ((\nu a \cdot P, e)\Gamma, L) \leadsto_{\text{can}} ((P, e[a \mapsto a'])\Gamma, L \cup \{a'\}) \\ & ((P \mid Q, e)\Gamma, L) \leadsto_{\text{can}} ((P, e)(Q, e)\Gamma, L) \end{aligned}$$

where, in the second rule, $a' \in \mathcal{N} \setminus (L \cup \text{Codom}(e) \cup \text{FV}(\Gamma))$. One shows easily that, up to α -equivalence, this reduction relation is confluent, and it is clearly strongly normalizing. We denote by $\text{Can}(\Gamma, L)$ the normal form of the state (Γ, L) for this rewriting relation. Observe that if $(\Gamma, L) \leadsto_{\text{can}} (\Delta, M)$ then $\text{FV}(\Delta, M) \subseteq \text{FV}(\Gamma, L)$.

4.2.3 Transitions. Next, one defines a labeled transition system $\mathbb{S}_{\mathcal{L}}$. The objects of this system are labeled canonical states and the transitions, labeled by pairs of labels, are defined as follows.

$$\begin{aligned} & (([l]a(b_1 \dots b_n) \cdot P, e)(\overline{[m]}a'(b'_1 \dots b'_n) \cdot P', e')\Gamma, L) \\ & \xrightarrow{l\overline{m}} \text{Can}((P, e[b_1 \mapsto e'(b'_1), \dots, b_n \mapsto e'(b'_n)])(P', e')\Gamma, L) \end{aligned}$$

if $e(a) = e'(a')$. Observe that if $(\Gamma, L) \xrightarrow{l\overline{m}} (\Delta, M)$ then $\text{FV}(\Delta, M) \subseteq \text{FV}(\Gamma, L)$.

4.3 Relating the rewriting and the abstract machine approaches to the operational semantics of the π -calculus

We recall a more standard way of presenting the operational semantics of the π -calculus and outline its equivalence with the environment machine style we have chosen.

One defines first a structural equivalence relation between labeled π -terms, denoted as $\sim$. It is the least equivalence relation such that

$$\begin{aligned} & \text{nil} \mid P \sim P \\ & P \mid Q \sim Q \mid P \\ & (P \mid Q) \mid R \sim P \mid (Q \mid R) \\ & \nu a \cdot \nu b \cdot P \sim \nu b \cdot \nu a \cdot P \\ & \nu a \cdot \text{nil} \sim \text{nil} \\ & (\nu a \cdot P) \mid Q \sim \nu a \cdot (P \mid Q) \quad \text{if } a \notin \text{FV}(Q) \end{aligned}$$

Then one can define a labeled transition system, where the transitions are labeled by pairs of labels (as all the transition systems we consider in the present paper). This transition system is defined by the following rules:

$$\begin{array}{c}
\hline
[l]a(b_1, \dots, b_n) \cdot P_1 \mid \overline{[m]}a\langle c_1, \dots, c_n \rangle \cdot P_2 \xrightarrow{l\overline{m}} P_1[c_1, \dots, c_n/b_1, \dots, b_n] \mid P_2 \\
\hline
\frac{P_1 \sim P'_1 \quad P'_1 \xrightarrow{l\overline{m}} P'_2 \quad P'_2 \sim P_2}{P_1 \xrightarrow{l\overline{m}} P_2} \quad \frac{P \xrightarrow{l\overline{m}} P'}{P \mid Q \xrightarrow{l\overline{m}} P' \mid Q} \\
\hline
\frac{P \xrightarrow{l\overline{m}} P'}{\nu a \cdot P \xrightarrow{l\overline{m}} \nu a \cdot P'}
\end{array}$$

Last one defines a translation relation $\mathcal{T}$ between states and processes. We say that $P \mathcal{T} (\Gamma, L)$ if $\Gamma = (P_1, e_1) \cdots (P_n, e_n)$, $L = \{a_1, \dots, a_k\}$ and $P \sim \nu a_1 \dots a_k \cdot ((e_1 \cdot P_1 \mid \cdots) \mid e_n \cdot P_n)$.

Proposition 6 *For any process P , one has $P \mathcal{T} \text{Can}((P, e), \emptyset)$ where e is the partial identity function whose domain is $\text{FV}(P)$. Moreover, the relation $\mathcal{T}$ is a strong bisimulation.*

The proof is easy.

4.4 Translation of processes to differential interaction nets

Since we do not work up to associativity and commutativity of contraction and cocontraction, it does not make sense to define this translation as a function from processes to nets. For each repetition-free list of names $a_1, \dots, a_n$, we define a relation $\mathcal{I}_{a_1, \dots, a_n}$ from processes whose free names are contained in $\{a_1, \dots, a_n\}$ to simple nets t which have $2n+1$ free ports $a_1^t, a_1^o, \dots, a_n^t, a_n^o$ and $\mathbf{c}$ as in Figure 13(a). The additional port $\mathbf{c}$ will be used for controlling the sequentiality of the reduction, thanks to transistors. Reducing the translation of a process will be possible only when a boxed identity cell will be connected to its control port. This is completely similar to the additional control free name in the translation of the π -calculus in solos, in [LV03]⁹.

⁹ There is a simple interpretation of solo diagrams into differential interaction nets, which uses only our toolbox without promotion so that solo diagrams can be seen as an intermediate graphical language which can be implemented in the low level differential syntax. Our translation of the π -calculus results from an analysis and a simplification of the composed translation “ π -calculus $\rightarrow$ solo diagrams $\rightarrow$ differential nets”. The simplification results from some rewiring and from the use of the boxed identity cells which is easily replicable. The translation of solos into differential nets leads to cycles (which appear when a name is identified with itself) which are avoided in the present direct translation. Well behaved conditions on solos for avoiding such

It will be possible to check that, if P and P' are α -equivalent, then $P \mathcal{I}_{a_1, \dots, a_n} s$ iff $P' \mathcal{I}_{a_1, \dots, a_n} s$. We define now the translation relation, by induction on processes. And next we define the translation relation for states.

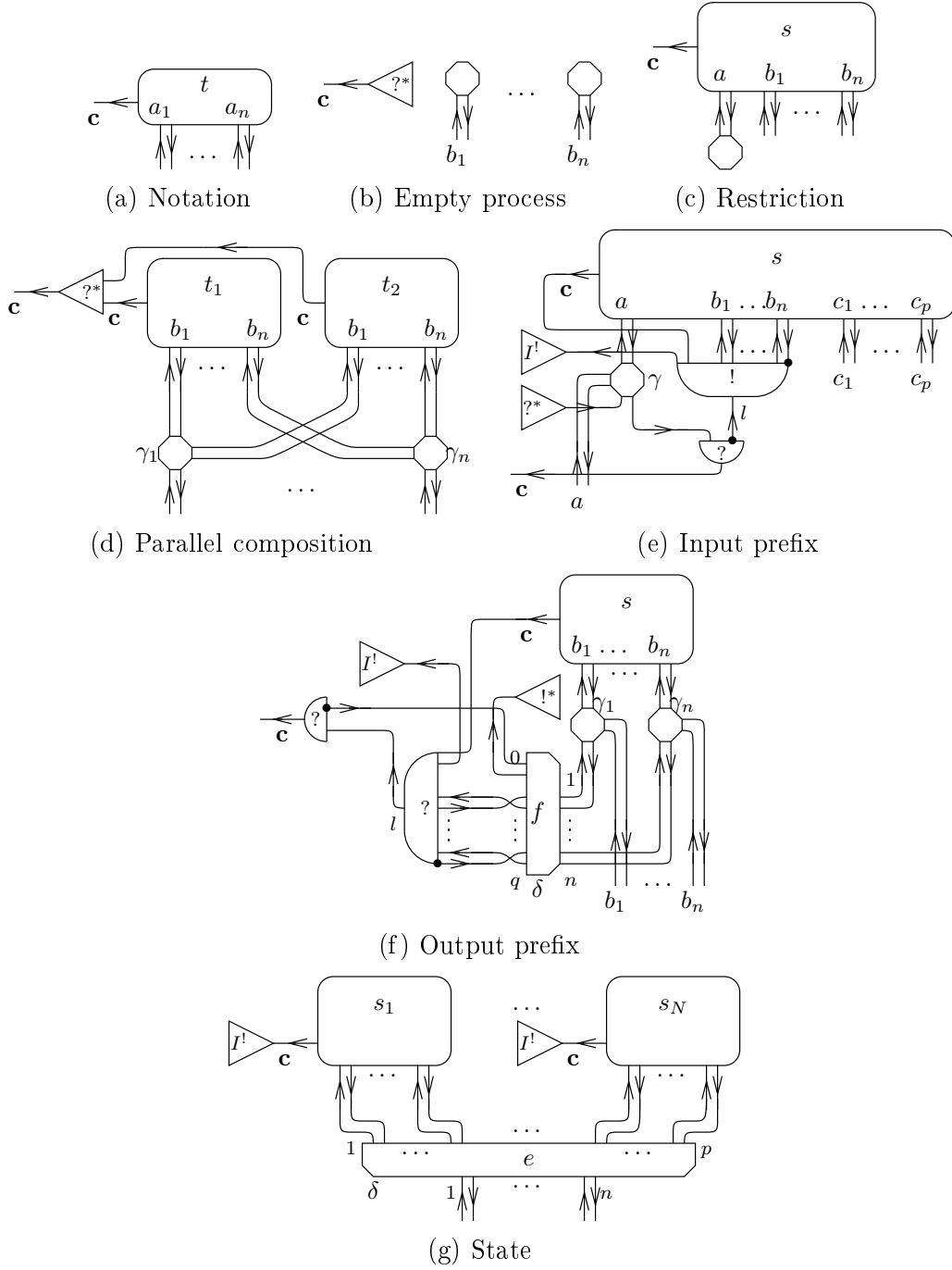


Fig. 13. Process and state translation

4.4.1 Empty process. One has $\text{nil } \mathcal{I}_{b_1, \dots, b_n} t$ if t is as in Figure 13(b).

cycles are introduced and studied in [EL07].

4.4.2 Name restriction. One has $\nu a \cdot P \mathcal{I}_{b_1, \dots, b_n} t$ iff t is as in Figure 13(c), with s satisfying $P \mathcal{I}_{a, b_1, \dots, b_n} s$.

4.4.3 Parallel composition. One has $P_1 \mid P_2 \mathcal{I}_{b_1, \dots, b_n} t$ iff the simple net t is as in Figure 13(d), where $P_1 \mathcal{I}_{b_1, \dots, b_n} t_1$, $P_2 \mathcal{I}_{b_1, \dots, b_n} t_2$ and $\gamma_1, \dots, \gamma_n$ are communication areas of order 1.

4.4.4 Input prefix. Let $l \in \mathcal{L}$. Assume that $a, b_1, \dots, b_n, c_1, \dots, c_p$ are pairwise distinct names and let $Q = [l]a(b_1 \dots b_n) \cdot P$. One has $Q \mathcal{I}_{a, c_1, \dots, c_p} t$ if t is as in Figure 13(e), where γ is a communication area of order 1 and where s is a simple net which satisfies $P \mathcal{I}_{a, b_1, \dots, b_n, c_1, \dots, c_p} s$.

4.4.5 Output prefix. Let $l \in \mathcal{L}$. Let $b_1, \dots, b_n$ be a list of pairwise distinct names and let $Q = \overline{[l]b_{f(0)}} \langle b_{f(1)} \dots b_{f(q)} \rangle \cdot P$, where $f : \{0, 1, \dots, q\} \rightarrow \{1, \dots, n\}$ is a function (this function is uniquely determined by Q and by the enumeration $b_1, \dots, b_n$). One has $Q \mathcal{I}_{b_1, \dots, b_n} t$ if t is as in Figure 13(f), where $\gamma_1, \dots, \gamma_n$ are communication areas of order 1, δ is an f -identification structure and where s is a simple net which satisfies $P \mathcal{I}_{b_1, \dots, b_n} s$.

4.4.6 States. Let $\Gamma = (P_1, e_1) \dots (P_N, e_N)$ be a soup and $b_1, \dots, b_n$ be a repetition-free list of names containing all the codomains of the environments $e_1, \dots, e_N$ (that is, containing $\text{FV}(\Gamma)$). We assume that the domains of the environments e_i are pairwise disjoint, which is possible up to α -equivalence. Let $a_1, \dots, a_p$ be a repetition-free enumeration of the elements of $\bigcup_{i=1}^N \text{Dom } e_i$, such that there is a list of non-negative integers $0 = h_0 \leq h_1 \leq \dots \leq h_N = p$ such that, for $i = 1, \dots, N$, the list $a_{h_{i-1}+1}, \dots, a_{h_i}$ is a repetition-free enumeration of the elements of $\text{Dom}(e_i)$. Let $e : \{1, \dots, p\} \rightarrow \{1, \dots, n\}$ be the map which is uniquely defined by the fact that, for each $i = 1, \dots, N$ and each j such that $h_{i-1} + 1 \leq j \leq h_i$, one has $e_i(a_j) = b_{e(j)}$.

Then one has $\Gamma \mathcal{I}_{b_1, \dots, b_n} t$ if t is a simple net of the shape shown in Figure 13(g), where $s_1, \dots, s_N$ are simple nets such that $P_i \mathcal{I}_{a_{h_{i-1}+1}, \dots, a_{h_i}} s_i$ and δ is an e -identification structure.

Last, if we are moreover given $L \subseteq \mathcal{N}$ and a repetition-free list of names $b_1, \dots, b_n$ containing all the free names of the state (Γ, L) , one has $(\Gamma, L) \mathcal{I}_{b_1, \dots, b_n} u$ if one has $\Gamma \mathcal{I}_{b_1, \dots, b_n, c_1, \dots, c_p} t$ for some repetition-free enumeration $c_1, \dots, c_p$ of L (assumed of course to be disjoint from $b_1, \dots, b_n$, which is always possible up to α -equivalence), and u is obtained by plugging communication areas of order -1 on the pairs of free ports of t corresponding to the c_j s.

A simple inspection of the translation above shows that, whenever $(\Gamma, L) \mathcal{I}_{b_1, \dots, b_n}$

u , the simple net u satisfies the CLB of 1.2.4.

5 Comparing the transition systems

5.1 A diving lemma

We first introduce the auxiliary notions of guarded cell and of a (co)dereliction cell diving into a process. We then state and prove two lemmata which will be crucial in the proofs of Propositions 10 and 11 which, in turn, will lead to Theorem 12.

5.1.1 Guarded dereliction and codereliction cells. Let $l, r \in \mathcal{L}$ be distinct, $r \neq \tau$ and let $s \in \Delta$. Let δ be a (co)dereliction cell labeled by l in s . One says that δ is *guarded by* (the dereliction or codereliction cell labeled by) r in s if there is a sequence $p_1, \dots, p_n$ of pairwise distinct ports of s such that

- p_1 is the auxiliary port of δ and p_2 is its principal port;
- p_{n-1} is the auxiliary port of r and p_n is its principal port;
- and for each i with $1 < i < n - 1$, either p_i and p_{i+1} are the two ports of a wire of s or there is a cell in s such that p_{i-1} is an auxiliary port of that cell and p_i is its principal port.

Such a sequence of ports will be called a *guarding path* from δ to r in s (observe that since $r \neq \tau$, there is no ambiguity on the (co)dereliction cell labeled by r in s , whereas l can be equal to τ and so there might be several (co)dereliction cells labeled by l in s).

5.1.2 Persistency.

Lemma 7 *Let s be a simple net, let $R \subseteq \mathcal{L}$, let l, r be labels which are distinct, with $r \neq \tau$. Let δ be an l -labeled (co)dereliction cell which is guarded by r in s and assume that $s \rightsquigarrow_R^* s_1 + \dots + s_p$ where the s_i are simple. Then δ and r occur, and δ is guarded by r , in each of the simple nets s_i .*

Proof. The proof is straightforward: the (co)dereliction r can take part only to non-deterministic reductions during an $\rightsquigarrow_R$ -reduction, and hence cannot disappear (more precisely, its only way of disappearing is by turning to 0 the whole simple net where it occurs). $\square$

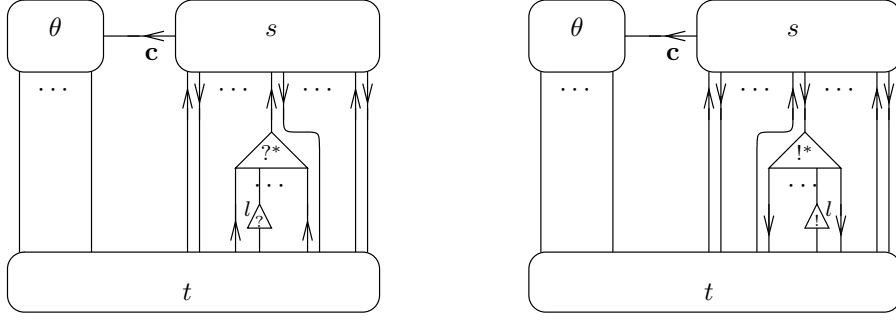


Fig. 14. Diving of dereliction and codereliction: initial configurations

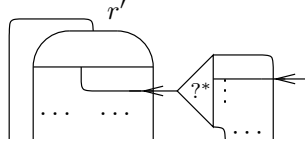


Fig. 15. Possible shape for the subnet θ of Figure 14

5.1.3 Diving of derelictions and coderelictions. Let $l \in \mathcal{L} \setminus \{\tau\}$, let u be a simple net, let P be a process. We say that l *dives into* P in u if there is a repetition-free list of names $b_1, \dots, b_n$ and a simple net s such that $P \mathcal{I}_{b_1, \dots, b_n} s$ and u is of one of the shapes (according to whether l labels a dereliction or a codereliction cell) shown in Figure 14, where θ is a boxed identity cell, or a net of the shape shown in Figure 15, consisting of a labeled input or output prefix compound cell, with a label $r' \neq \tau$.

With these notations, our aim is here to prove the following property.

Lemma 8 (Diving) *Assume that $l \in \mathcal{L} \setminus \{\tau\}$ dives into P in the simple net u , and let $m \in \mathcal{L} \setminus \{\tau\}$ which does not occur in P . Then u is $\{l, m\}$ -neutral.*

The label m cannot occur in P , but it can occur in the remainder of u ; the meaning of the lemma is that, during the reduction, “ l cannot exit from P ” or, more precisely, if it exits, it is by the control port c .

Proof. By induction on P (and, in some cases, by contradiction: in these cases, we assume that $u \rightsquigarrow_{\{l, m\}}^* u_1 + U$ and that u_1 contains an (l, m) -communication redex).

Assume first that $P = \text{nil}$. Assume that l is a dereliction. Then u has the shape of figure 16. Thus $u \rightsquigarrow_{\{l, m\}}^* 0$ by 3.3.5. Hence by the Church-Rosser property of $\rightsquigarrow_{\{l, m\}}^*$, we must have $u_1 + U \rightsquigarrow_{\{l, m\}}^* 0$. But this is impossible by Lemma 3 since u_1 has an (l, m) -communication redex. The case where l is a codereliction is similar.

The case $P = P_1 \mid P_2$ is similarly handled: using 3.3.5 and the inductive

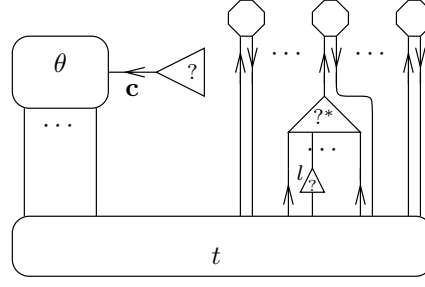
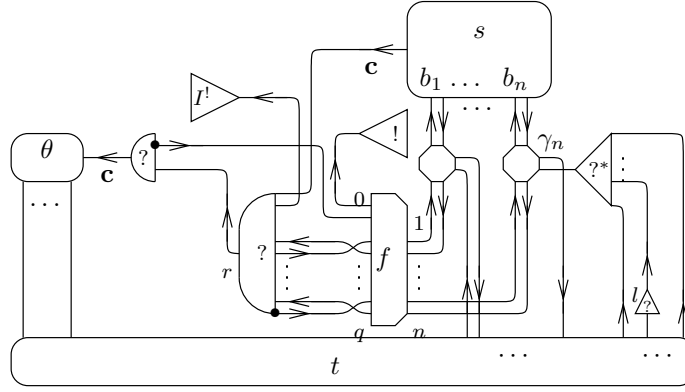


Fig. 16. Diving lemma

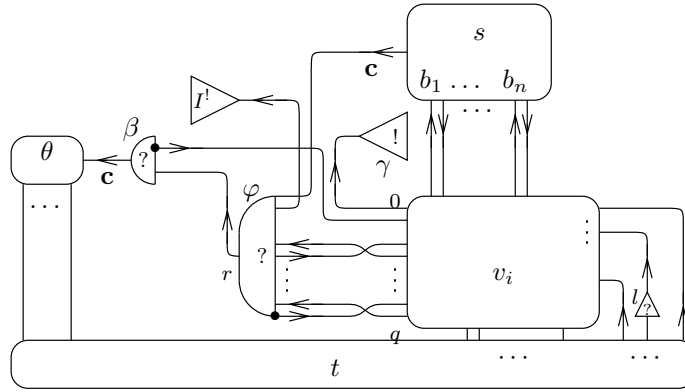
hypothesis, one shows that $u \rightsquigarrow_{\{l,m\}}^* V$ where V is a sum of $\{l,m\}$ -neutral simple nets, and hence u is $\{l,m\}$ -neutral by Lemma 4.

If $P = \nu a \cdot Q$, one applies directly the inductive hypothesis.

To conclude, we consider the case where $P = \overline{[r]b_{f(0)}\langle b_{f(1)} \dots b_{f(p)} \rangle} \cdot Q$. Assume first that l is a dereliction. Then u is of the following shape (without loss of generality, we assume that the dereliction is connected to a port corresponding to the name b_n), where s is a simple net satisfying $Q \mathcal{I}_{b_1, \dots, b_n}$ s:



Then, aggregating first the communication area γ_n with the communication area of the f -identification structure to which it is connected, we see that we have $u \rightsquigarrow_{\{l,m\}}^* \sum_{i=1}^N u_i$ where u_i is a simple net which has the following shape

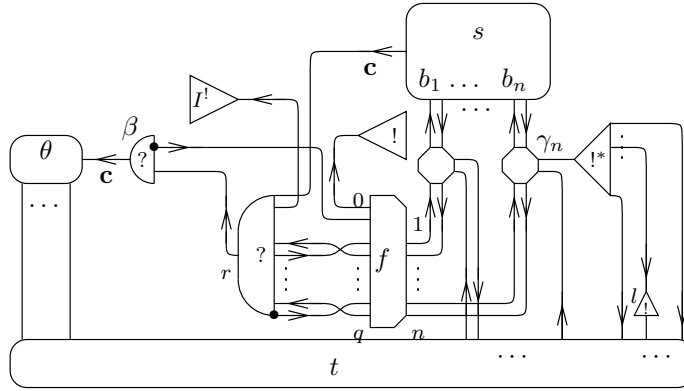


where, according to 3.3.5, in v_i , the principal port of l is forwarded (see the definition of this concept in 3.3.3)

- (1) to the port b_n^+ of s
- (2) or to the principal port of the coweakening cell γ , in the case where $f(0) = n$
- (3) or to one of the input auxiliary port of the compound cell φ , corresponding to an index $j \in \{1, \dots, q\}$ such that $f(j) = n$.

For i satisfying (2), we have $u_i \rightsquigarrow_{\{l, m\}}^* 0$. For i satisfying (3), l is guarded by $r \neq \tau$ (the labeled dereliction cell of φ) in u_i , and so u_i is $\{l, m\}$ -neutral by Lemma 7. For i satisfying (1), the inductive hypothesis applies, showing that u_i is $\{l, m\}$ -neutral. Therefore u is $\{l, m\}$ -neutral by Lemma 4.

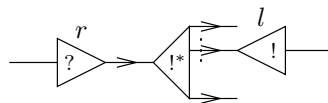
Assume now that l is a codereliction, so that u has the following shape (with the same notations as above).



As before, we have $u \rightsquigarrow_{\{l, m\}}^* \sum_{i=1}^N u_i$ where the u_i s have the same shape as before. Using the same notations, in v_i , the principal port of l is forwarded

- (1) to the port b_n^- of s
- (2) or to the dotted auxiliary port of the transistor output compound cell β , in the case where $f(0) = n$
- (3) or to one of the input auxiliary port of the compound cell φ , corresponding to an index $j \in \{1, \dots, q\}$ such that $f(j) = n$.

The cases (1) and (3) are handled as before. So consider an index i corresponding to case (2). There are two possibilities, depending on the value of the net θ . If θ is a boxed identity cell, then $u_i \rightsquigarrow_{\{l, m\}}^* u'$ where u' is a simple net which contains the following subnet



Since we have $r \notin \{l, m\}$ (remember that we have assumed that m does not occur in P), this subnet has no $\sim_{\{l, m\}}^*$ -redex, and therefore, it will still be present in any simple summand of a net u'' such that $u' \sim_{\{l, m\}}^* u''$. So u' is $\{l, m\}$ -neutral, and so is u by Lemma 4.

Assume last that θ consists of an r' -labeled output or input prefix compound cell (with $r' \neq \tau$) together with a generalized contraction cell (second possibility for θ in 5.1.3, see Figure 15). Here we can have $r' = m$, but l is guarded by r' in u , and hence u is $\{l, m\}$ -neutral by Lemma 7 and Lemma 4.

The case where P starts with an input prefix is completely similar to that of an output prefix, and of course simpler. $\square$

Lemma 9 *Let (Γ, L) be a state and let $b_1, \dots, b_n$ be a repetition-free enumeration of the free names of (Γ, L) . Let (Δ, M) be the canonical form of (Γ, L) and let s be a simple net such that $(\Gamma, L) \mathcal{I}_{b_1, \dots, b_n} s$. Then there exists a simple net t such that $(\Delta, M) \mathcal{I}_{b_1, \dots, b_n} t$ and $s \sim_s t$.*

The equivalence relation $\sim_s$ is defined in 2.1.4. The proof is by simple inspection of the definition of the interpretation relation, using 3.3.1.

We establish now two results which are the main ingredients towards our bisimulation theorem.

Proposition 10 *Let (Γ, L) and (Δ, M) be canonical states and let $l, m \in \mathcal{L} \setminus \{\tau\}$. Assume that $(\Gamma, L) \xrightarrow{\overline{lm}} (\Delta, M)$. Let s be a simple net such that $(\Gamma, L) \mathcal{I}_{b_1, \dots, b_n} s$ where $b_1, \dots, b_n$ is a repetition-free list of names containing all the free names of (Γ, L) . Then there are simple nets t_0 and t such that $(\Delta, M) \mathcal{I}_{b_1, \dots, b_n} t$, $s \xrightarrow{\overline{lm}} t_0$ and $t_0 \sim_d t$.*

Proof. We know that Γ must be of the shape

$$([l]a(c_1 \dots c_p) \cdot P, e_1)(\overline{[m]d_{f(0)}} \langle d_{f(1)} \dots d_{f(p)} \rangle \cdot Q, e_2)(P_3, e_3) \dots (P_N, e_N) \quad (1)$$

where we assume that the e_i s have pairwise disjoint domains, that $a, c_{p+1}, \dots, c_{p+q}$ is a repetition-free enumeration of the domain of e_1 (these names are assumed to be distinct from the names $c_1, \dots, c_p$, which are bound in the first process of the soup (1)), that $d_1, \dots, d_r$ is a repetition-free enumeration of the domain of e_2 , that $h_1, \dots, h_m$ is a repetition-free enumeration of the union of the domains of $e_3, \dots, e_N$, and $f : \{0, \dots, p\} \rightarrow \{1, \dots, r\}$ is a function, and we have $e_1(a) = e_2(d_{f(0)})$. And $(\Delta, M) = \text{Can}(\Gamma', L)$ where

$$\Gamma' = (P, e_1[c_1 \mapsto e_2(d_{f(1)}), \dots, c_p \mapsto e_2(d_{f(p)})])(Q, e_2)(P_3, e_3) \dots (P_N, e_N).$$

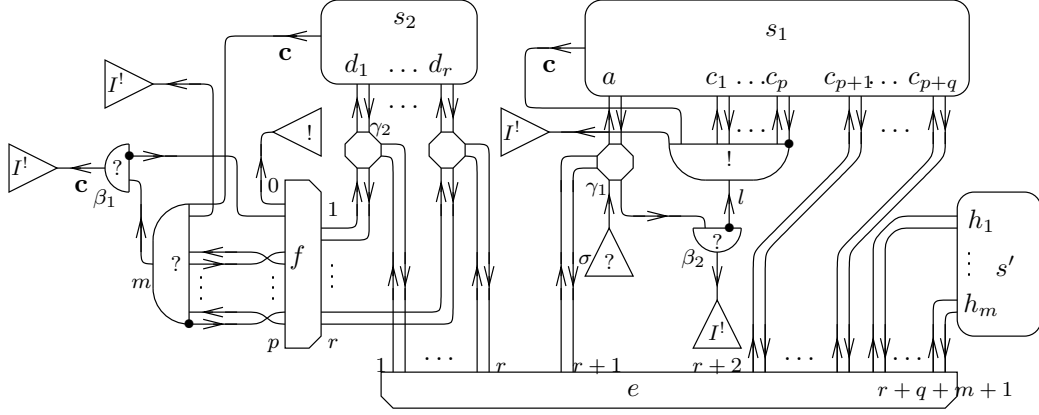


Fig. 17. Translation of state of Formula (1)

Without loss of generality, we can assume that $f(0) = 1$. With these notations, the simple net s is of the shape shown in Figure 17, where s_1 is a simple net such that $P \mathcal{I}_{a, c_1, \dots, c_{p+q}} s_1$, s_2 is a simple net such that $Q \mathcal{I}_{d_1, \dots, d_r} s_2$ and s' stands for the juxtaposition of simple nets s_i s such that $P_i \mathcal{I}_{\vec{h}^i} s_i$ (for $3 \leq i \leq N$) where $\vec{h}^i$ stands for an enumeration of the domain of e_i (so that the lists of names $\vec{h}^i$ are pairwise disjoint, and their concatenation is a repetition-free enumeration of the names $h_1, \dots, h_m$), with a boxed identity connected to the control ports of each s_i . In this net, e is the function $\{1, \dots, r+q+m+1\} \rightarrow \{1, \dots, n\}$ which corresponds to the union of the functions e_i for $i = 1, \dots, N$. Observe that we have $e(1) = e(r+1)$ since by hypothesis $e_1(a) = e_2(d_1)$.

We have omitted in Figure 17 the pairs of free ports corresponding to $b_1, \dots, b_n, b_{n+1}, \dots, b_{n+n'}$, the names b_i for $i > n$ corresponding to the elements of L ; remember that they are there and that each pair of free port corresponding to a b_i with $i > n$ is connected to a communication area of order -1 .

Then we can reduce the net of Figure 17 along the following steps.

- Observe first that the pairs of ports 1 and $r+1$ (attached to the domain of e) are connected to a common communication area δ_1 in the identification structure labeled by e (see 3.2.2) since $e(1) = e(r+1)$, and also that the codomain pair of ports 1 and the domain pair of ports 0 of the identification structure labeled by f are connected to a common communication area δ_2 in this identification structure, since $f(0) = 1$. We apply reduction 3.3.1 for aggregating the communication areas $\gamma_1, \delta_1, \gamma_2$ and δ_2 in a unique communication area δ . Let u be the obtained simple net, we have $s \rightsquigarrow_{\{l, m\}}^* u$.
- Apply reduction 3.3.7 to both transistors β_1 and β_2 and let u' be the obtained simple net, we have $u \rightsquigarrow_{\{l, m\}}^* u'$.
- u' contains therefore subnet v shown in Figure 18 where, for $i = -1, 0, \dots, g$ the pair of ports (r_{2i+3}, r_{2i+4}) is connected either
 - (1) to the pair of port a of s_1
 - (2) or to one of the pairs of ports $c_{p+1}, \dots, c_{p+q}$ of s_1

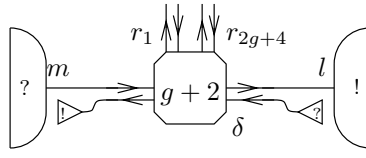


Fig. 18. The subnet v

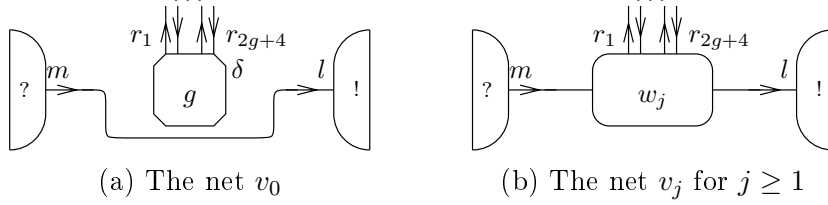
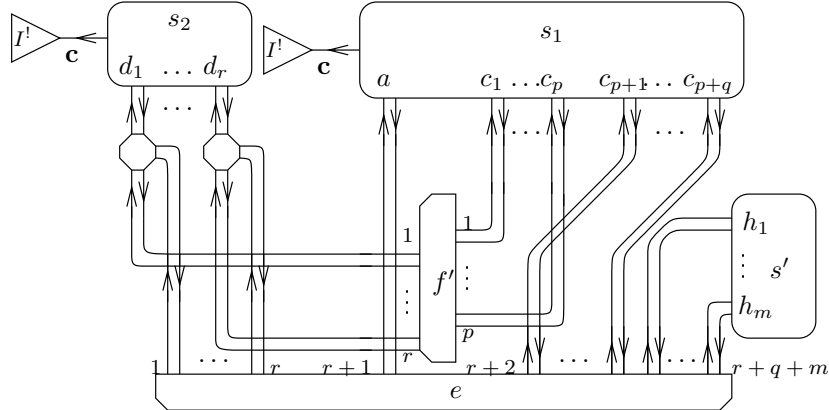


Fig. 19. Reducing v

- (3) or to one of the pairs of ports $h_1, \dots, h_m$ of s'
- (4) or to a pair of ports of one of the communication areas connected to $d_2, \dots, d_r$
- (5) or to the pair of ports d_1
- (6) or to one of the auxiliary pairs of ports of the output prefix compound cell labeled by m
- (7) or to one of the pairs of ports b_h corresponding to codomain pairs of ports of the identification structure e ; these pairs of ports are either free in s (and hence in u') or connected to a communication area of order -1 .

To v , we can apply reduction 3.3.4. This subnet reduces by the $\sim_{\{l,m\}}^*$ reduction to a sum $v_0 + v_1 + \dots + v_k$ where v_0 is shown in Figure 19(a) and the v_j s ($j \geq 1$) are nets of the shape shown in Figure 19(b) where the principal port of l and m are forwarded to ports among $r_1, \dots, r_{2g+4}$. We have $u' \sim_{\{l,m\}}^* u'_0 + u'_1 + \dots + u'_k$ where u'_j is obtained by replacing in u' the net v by the net v_j ($j = 0, \dots, k$).

- We apply the (l, m) -communication reduction to u'_0 , getting a simple net t_0 which is $\sim_d$ equivalent to the following simple net

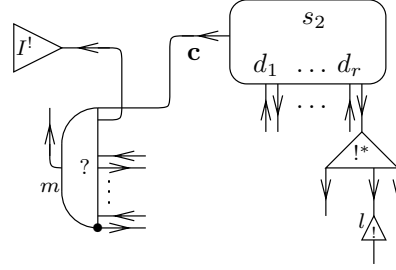


where f' is the restriction of f to $\{1, \dots, p\}$. This net is $\sim_s$ equivalent to a simple net t_1 with $(\Gamma', L) \mathcal{I}_{b_1, \dots, b_n} t_1$ (upon applying 3.3.1 to the communi-

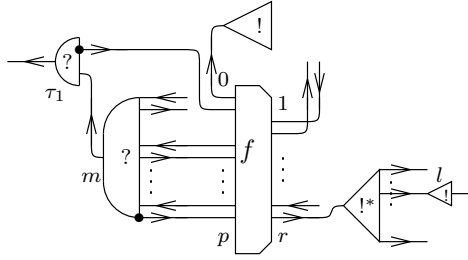
cation areas of the identification structure f' , the ones which are connected to the pairs of free ports d_i of s_2 and those belonging to the identification structure e). By Lemma 9, there is a simple net t such that $t_1 \sim_s t$ and $(\Delta, M) \mathcal{I}_{b_1, \dots, b_n} t$.

To conclude, we must check that, for $j \geq 1$, u'_j is $\{l, m\}$ -neutral. But, for each of the two labels l and m , we are in one of the seven cases (1) to (7) above. Consider for instance label l . If we are in case (1), (2), (3), (5), we can directly apply Lemma 8.

Assume that we are in case (4), we can apply 3.3.5 and see that $u'_j \rightsquigarrow_{\{l, m\}}^* w_1 + w_2$ where w_1 and w_2 are simple, and w_1 contains a subnet of the shape shown beside (assuming that in u'_j , l is forwarded to the communication area connected to d_r). Hence by Lemma 8, w_1 is $\{l, m\}$ -neutral.



On the other hand, w_2 contains a subnet of the shape beside. This subnet $\rightsquigarrow_{\{l, m\}}^*$ reduces by 3.3.5 to a sum of simple nets in each of which l is guarded by m . Therefore, by Lemmata 7 and 4, w_2 is $\{l, m\}$ -neutral. So by the same lemma, u'_j is $\{l, m\}$ -neutral.

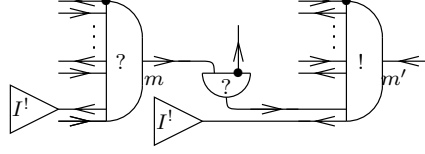


If we are in case (6) then, in u'_j , l is guarded by m and hence u'_j is $\{l, m\}$ -neutral by Lemma 7. Last assume we are in case (7); in this case, l is connected to an auxiliary port of a generalized structural cell whose principal port is free, or is connected to a weakening cell. In both cases again it is clear that u'_j is $\{l, m\}$ -neutral $\square$

Proposition 11 *Let (Γ, L) be a canonical state and $b_1, \dots, b_n$ be a repetition-free list of names containing all the free names of (Γ, L) . Let s be a simple net such that $(\Gamma, L) \mathcal{I}_{b_1, \dots, b_n} s$. If t'_0 is a simple net such that $s \xrightarrow{l\overline{m}} t'_0$, then there is a canonical state (Δ, M) such that $(\Gamma, L) \xrightarrow{l\overline{m}} (\Delta, M)$ and there exists a simple net t such that $(\Delta, M) \mathcal{I}_{b_1, \dots, b_n} t$ and $t \sim_d t'_0$.*

Proof. One shows first that both l and m must be minimal in the poset $\mathcal{L}(\Gamma, L)$ (see Section 4.2). Assume for instance that m is not minimal. Then the principal port of the dereliction cell labeled by m is connected to an auxiliary port

of a transistor whose principal port is connected to an auxiliary port of an input or output prefix cell, labeled say by m' , with $m' < m$ (actually, m' is the predecessor of m in the forest $\mathcal{L}(\Gamma, L)$). Say for instance that the prefix cell labeled by m' is an input prefix cell. So s contains the following subnet



So m is guarded by m' in s and so, whenever $s \rightsquigarrow_{\{l,m\}}^* s'$, no simple net appearing in s' can contain an (l, m) -communication redex, in contradiction with our hypothesis that $s \xrightarrow{l\bar{m}} t'_0$.

We have seen that l and m are minimal in the poset $\mathcal{L}(\Gamma, L)$ and this means that in Γ , the prefixes labeled by l and m are the outermost prefixes of P_1 and P_2 where $\Gamma = (P_1, e_1) \cdots (P_N, e_N)$ (and the choice of P_1 and P_2 is uniquely determined by l and m), that is, Γ is of the form described by Formula (1) in the proof of Proposition 10, P_1 denoting the first process in that expression, which is guarded by an l -labeled input prefix, and P_2 the second one, which is guarded by an m -labeled output prefix. Using the notations of Formula (1), we argue now that necessarily $e_1(a) = e_2(d_{f(0)})$ (we can refer to Figure 17 as describing s). But if this is not the case, an inspection of the interpretation of input prefixes 4.4.4, of states 4.4.6 and of the identification structure associated with the “global environment” e (see 3.2.2) shows that $s \rightsquigarrow_{\{l,m\}}^* s' = s'_1 + \cdots + s'_q$ where for each i , s'_i is simple and one of the following holds:

- (1) in s'_i , l is forwarded to a free port of s'
- (2) or in s'_i , l dives into a subnet t such that $P_j \mathcal{I}_{c_1, \dots, c_r} t$ for some $j = 1, \dots, N$ and $c_1, \dots, c_r$ is a repetition-free enumeration of the domain of e_j .

In case (1), s'_i is $\{l, m\}$ -neutral. The same is true of s'_i in case (2) when the index j is different from 2 since then P_j cannot contain the label m and we can apply Lemma 8. In the case $j = 2$, using our assumption that $e_1(a) \neq e_2(d_{f(0)})$, we see that l dives into t through a free port which does not correspond to $d_{f(0)}$ and from this (and from an inspection of the interpretation of output prefixes 4.4.5), we see that $s_i \rightsquigarrow_{\{l,m\}}^* s'$ where s' is a sum of simple nets in which, either l is guarded by m , or l dives into a subnet u of t such that $Q \mathcal{I}_{h_1, \dots, h_q} u$ (for a suitable list of names $h_1, \dots, h_q$), where Q is the process guarded by the m -labeled output prefix of P_2 (and therefore, Q does not contain the label m). Applying Lemma 7 in the first case and Lemma 8 in the second case, we see that each simple summand of s' is $\{l, m\}$ -neutral and therefore s_i also is $\{l, m\}$ -neutral by Lemma 4. Finally, by the same lemma, s itself is $\{l, m\}$ -neutral, contradicting the hypothesis that $s \xrightarrow{l\bar{m}} t'_0$.

So we must have $e_1(a) = e_2(d_{f(0)})$ and since our processes and states are implicitly arity-typed (see 4.1), we know that the number of objects of the two involved prefixes coincide (the common value of these numbers is p , according to our notations).

Using the same notations as in Proposition 10, and the statement itself of this theorem, we have $(\Gamma, L) \xrightarrow{l\overline{m}} (\Delta, M)$ and there are simple nets t and t_0 such that $(\Delta, M) \mathcal{I}_{b_1, \dots, b_n} t$, $t \sim_d t_0$ and $s \xrightarrow{l\overline{m}} t_0$. This means more precisely that $s \rightsquigarrow_{\{l, m\}}^* s' = s_0 + s_1 + \dots + s_p$, with the s_j s simple, such that s_0 has an (l, m) -communication redex and each s_j (for $j \geq 1$) is $\{l, m\}$ -neutral and t_0 is the net obtained by reducing the (l, m) -communication redex of s_0 .

We conclude by showing that $t_0 \sim_d t'_0$.

We know from our hypothesis that $s \rightsquigarrow_{\{l, m\}}^* s'' = s'_0 + s'_1 + \dots + s'_q$, where s'_0 has an (l, m) -communication redex and each s'_j (for $j \geq 1$) is $\{l, m\}$ -neutral, and t'_0 is the simple net obtained from s'_0 by reducing its (l, m) -communication redex.

By the Church Rosser property of $\rightsquigarrow_{\{l, m\}}^*$, there is a net u such that $s' \rightsquigarrow_{\{l, m\}}^* u$ and $s'' \rightsquigarrow_{\{l, m\}}^* u$. By Lemma 3, we have $u = u_0 + u'$ with $s_0 \rightsquigarrow_d^* u_0$ and $s'_0 \rightsquigarrow_d^* u_0$, thanks also to the $\{l, m\}$ -neutrality of s_j and s'_j for $j \geq 1$. Moreover (still by Lemma 3), u_0 contains an (l, m) -communication redex as well, and if v_0 is the net obtained by reducing the (l, m) -communication redex of u_0 , we have also $t_0 \rightsquigarrow_d^* v_0$ and $t'_0 \rightsquigarrow_d^* v_0$. So we have $t_0 \sim_d t'_0$. $\square$

We are now ready to state a bisimulation theorem. Given a repetition-free list $b_1, \dots, b_n$ of names, we define a relation $\tilde{\mathcal{I}}_{b_1, \dots, b_n}$ between states and simple nets by: $(\Gamma, L) \tilde{\mathcal{I}}_{b_1, \dots, b_n} s$ if there exists a simple net s_0 such that $(\Gamma, L) \mathcal{I}_{b_1, \dots, b_n} s_0$ and $s_0 \sim_d s$.

Theorem 12 *The relation $\tilde{\mathcal{I}}_{b_1, \dots, b_n}$ is a strong bisimulation between the labeled transition systems $\mathbb{S}_{\mathcal{L}}$ and $\mathbb{D}_{\mathcal{L}}$.*

Proof. Let (Γ, L) be a canonical state and s_1 be a simple net, and assume that $(\Gamma, L) \tilde{\mathcal{I}}_{b_1, \dots, b_n} s_1$. So there is a simple net s such that $(\Gamma, L) \mathcal{I}_{b_1, \dots, b_n} s$ and $s \sim_d s_1$.

Assume first that $(\Gamma, L) \xrightarrow{l\overline{m}} (\Delta, M)$, with l, m two distinct elements of $\mathcal{L} \setminus \{\tau\}$. By Proposition 10, there are simple nets t_0 and t such that $(\Delta, M) \mathcal{I}_{b_1, \dots, b_n} t$, $t_0 \sim_d t$ and $s \xrightarrow{l\overline{m}} t_0$. By Lemma 5 ($\sim_d$ is a bisimulation), there exists t_1 such that $t \sim_d t_1$ and $s_1 \xrightarrow{l\overline{m}} t_1$. We have $(\Delta, M) \tilde{\mathcal{I}}_{b_1, \dots, b_n} t_1$.

Conversely, assume that $s_1 \xrightarrow{l\overline{m}} t_1$. By Lemma 5, there exists t such that

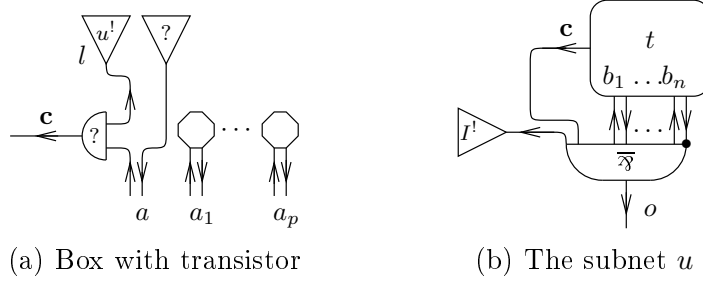


Fig. 20. Translation of input replication

$t \sim_d t_1$ and $s \xrightarrow{l\overline{m}} t$. By Proposition 11, there is a canonical state (Δ, M) and a simple net t_0 such that $(\Gamma, L) \xrightarrow{l\overline{m}} (\Delta, M)$ and $(\Delta, M) \mathcal{I}_{b_1, \dots, b_n} t_0 \sim_d t$. We have $(\Delta, M) \tilde{\mathcal{I}}_{b_1, \dots, b_n} t_1$. $\square$

6 Dealing with replication

We extend our π -calculus with the following construction: if $l \in \mathcal{L}$, if a and $b_1, \dots, b_n$ are pairwise distinct names and if P is a process such that $\text{FV}(P) \subseteq \{b_1, \dots, b_n\}$, then $[l]!a(b_1, \dots, b_n) \cdot P$ is a process, whose only free name is a . This process is guarded, in the sense of 4.2.2. This extension has no influence on the definition of the relation $\sim_{\text{can}}$ on the states of our environment machine. The transition of the machine has to be extended with the following rule:

$$\begin{aligned}
 & (([l]!a(b_1 \dots b_n) \cdot P, e)(\overline{[m]}a' \langle b'_1 \dots b'_n \rangle \cdot P', e') \Gamma, L) \\
 & \xrightarrow{l\overline{m}} \text{Can}([l]!a(b_1 \dots b_n) \cdot P, e) \\
 & \quad (P, e[b_1 \mapsto e'(b'_1), \dots, b_n \mapsto e'(b'_n)])(P', e') \Gamma, L)
 \end{aligned}$$

We extend now the translation relation $\mathcal{I}$ to the replicated input process (with the already mentioned closeness restriction). Given a process P whose free names are contained in the repetition-free list $b_1, \dots, b_n$, and given a list $a, a_1, \dots, a_p$ of pairwise distinct variables, we set $[l]!a(b_1 \dots b_n) \cdot P \mathcal{I}_{a, a_1, \dots, a_p} s$ if, for some simple net t such that $P \mathcal{I}_{b_1, \dots, b_n} t$, s is of the shape given by Figure 20(a). The promotion cell of that net contains the net shown in Figure 20(b).

When $P \mathcal{I}_{b_1, \dots, b_n} t$ for a process with replication P , the simple net t does not satisfy the CLB (see 1.2.4) in general since promotion cells will have labels $\neq \tau$, so that a bisimulation theorem will be harder to obtain (the transition system of simple nets is defined only for nets satisfying the CLB in Section 2.3). Simulation of processes by simple nets seems to be at hand: it suffices to define a more “liberal” transition system on simple nets. A natural candidate

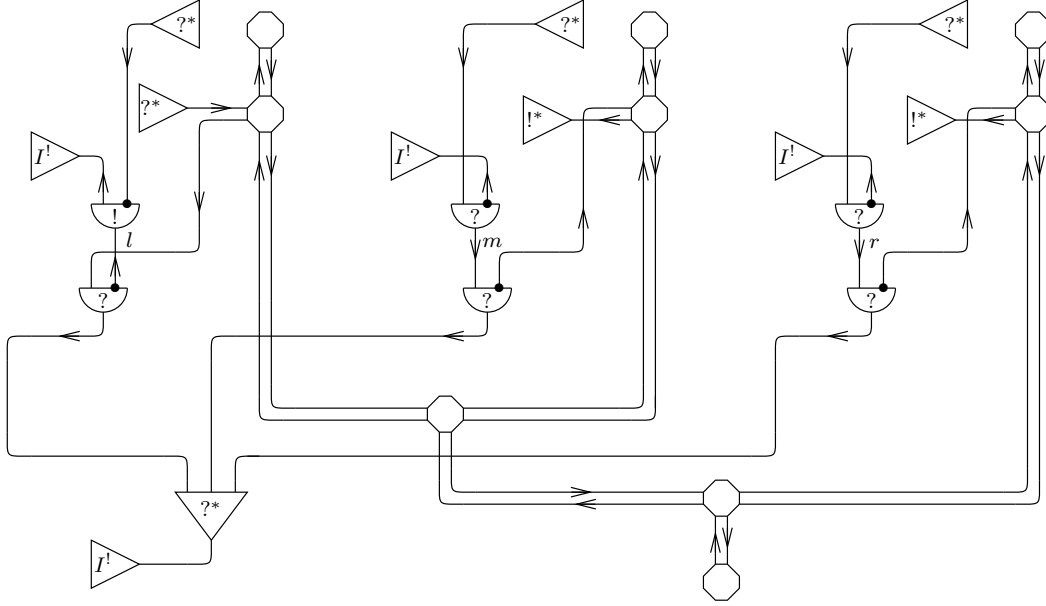


Fig. 21. Concurrent communication in a CCS process

for such a transition system is: $s \xrightarrow{\overline{lm}} t$ if $s \rightsquigarrow_{\{l,m\}}^* s_1 + s_2 + \dots + s_n$ where s_1 is a simple net which contains an (l, m) -communication or an (l, m) -box redex and becomes t when reducing this redex (no neutrality assumption on the t_i s for $i \geq 2$). We conjecture that our translation is a simulation from $\mathbb{S}_{\mathcal{L}}$ to this new transition system.

For obtaining more precise results, one should label in a different way the various copies of promotion cells, in the spirit of the geometry of interaction [Gir88a], with a similar discipline for processes as well.

7 Examples

We give a few examples to illustrate some key features of communication in the π -calculus as represented in differential interaction nets.

7.1 Concurrent communication

Let P be the process:

$$\nu a \cdot \left(([l]a() \cdot \text{nil} \mid \overline{[m]}a\langle \rangle \cdot \text{nil}) \mid \overline{[r]}a\langle \rangle \cdot \text{nil} \right).$$

The simplest state containing P is $(\Gamma, L) = ((P, \emptyset), \emptyset)$. We have $(\Gamma, L) \mathcal{I} s$ where s is the simple net of Figure 21.

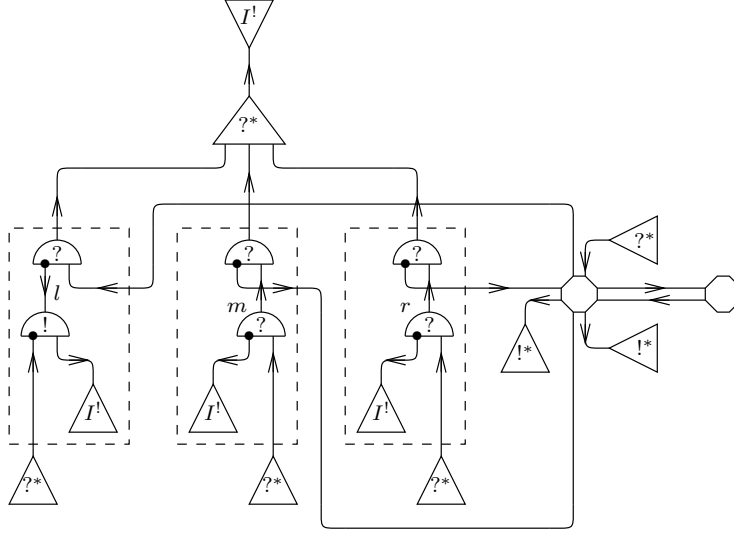


Fig. 22. A CCS process: first step

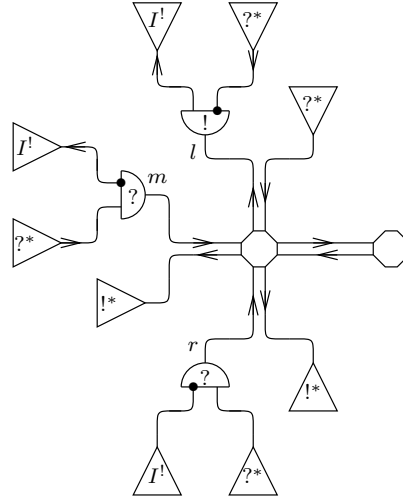


Fig. 23. A CSS process: second step

By applying aggregations of communication areas, we obtain the simple net s_1 of Figure 22. Thus $s \rightsquigarrow_s^* s_1$. Since P is in fact a CCS process (namely $\nu a \cdot (a \mid \bar{a} \mid \bar{a})$), we can remark how the translation into differential interaction nets is given by first a tree (with nodes represented with dashed boxes) corresponding to the tree structure of the CCS process (built from sequential and parallel compositions), and second communication areas for the identification of names.

The simple net s_1 reduces to the net s_2 ($s_1 \rightsquigarrow_d^* s_2$) of Figure 23, where the choice between actions ready to communicate will be done. This means that s_2 reduces to a sum of simple nets containing in particular the net s_3 ($s_2 \rightsquigarrow_{\{l,m\}}^* s_3 + \dots$) of Figure 24. If t is obtained from s_3 by reducing the (l, m) -communication redex, we have $s \xrightarrow{l\bar{m}} t$. This corresponds to $(\Gamma, L) \rightsquigarrow_{\text{can}}$

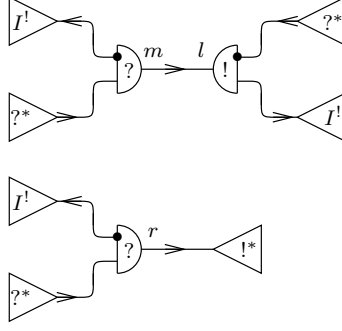


Fig. 24. A CSS process: final state

$(([l]a() \cdot \text{nil}, e)(\overline{[m]}a\langle \rangle \cdot \text{nil}, e)(\overline{[r]}a\langle \rangle \cdot \text{nil}, e), \{a'\}) \xrightarrow{\overline{lm}} ((\overline{[r]}a\langle \rangle \cdot \text{nil}, e), \{a'\})$ (with e defined only on $\{a\}$ by $e(a) = a'$) in the environment machine.

7.2 Sequentiality

Let P be the process:

$$[l]a() \cdot [l']b() \cdot \text{nil} \mid \overline{[m']}b\langle \rangle \cdot \text{nil} \mid \overline{[m]}a\langle \rangle \cdot \text{nil}$$

The simplest state containing P is $(\Gamma, L) = ((P, e), \emptyset)$ (with e defined on $\{a, b\}$ by $e(a) = a'$ and $e(b) = b'$). We have $(\Gamma, L) \mathcal{I}_{a', b'}$ s with $s \rightsquigarrow_s^* s_1$ (aggregations of communication areas) and s_1 is the simple net of Figure 25. Since P is again a CCS process (namely $a \cdot b \mid \overline{b} \mid \overline{a}$), we can see its tree structure in the differential interaction net s_1 of Figure 25.

The simple net s_1 reduces to the simple net s_2 of Figure 26 ($s_1 \rightsquigarrow_d^* s_2$).

Then there exists a simple net s_3 such that $s_2 \rightsquigarrow_{\{l, m\}}^* s_3 + \dots$ and if t is obtained from s_3 by reducing the (l, m) -communication redex it contains, we have $s \xrightarrow{\overline{lm}} t$. Moreover t reduces to the net of Figure 27. This corresponds to $(\Gamma, L) \rightsquigarrow_{\text{can}} (([l]a() \cdot [l']b() \cdot \text{nil}, e)(\overline{[m']}b\langle \rangle \cdot \text{nil}, e)(\overline{[m]}a\langle \rangle \cdot \text{nil}, e), \emptyset) \xrightarrow{\overline{lm}} (([l']b() \cdot \text{nil}, e)(\overline{[m']}b\langle \rangle \cdot \text{nil}, e), \emptyset)$ in the environment machine.

7.3 Name passing

Let P , Q and R be processes such that the free names of P are a and z , the only free name of Q is y and the free names of R are x and b . Let P' be the process:

$$\nu z \cdot (\overline{[l]}a\langle z \rangle \cdot P \mid [l']z(y) \cdot Q) \mid [m]a(x) \cdot \overline{[m']}x\langle b \rangle \cdot R$$

The simplest state containing P' is $(\Gamma, L) = ((P', e), \emptyset)$ (with e defined on $\{a, b\}$ by $e(a) = a'$ and $e(b) = b'$). If $P \mathcal{I}_{a, z} s_1$, $Q \mathcal{I}_y s_2$ and $R \mathcal{I}_{x, b} s_3$, we have

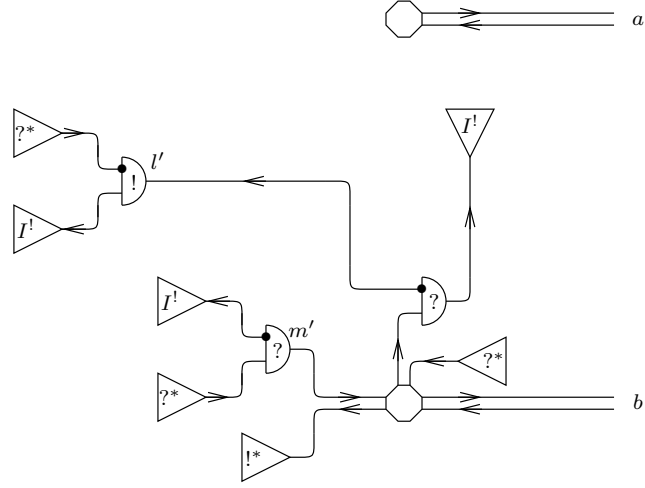


Fig. 27. Sequentiality: simple net s_3 , second step

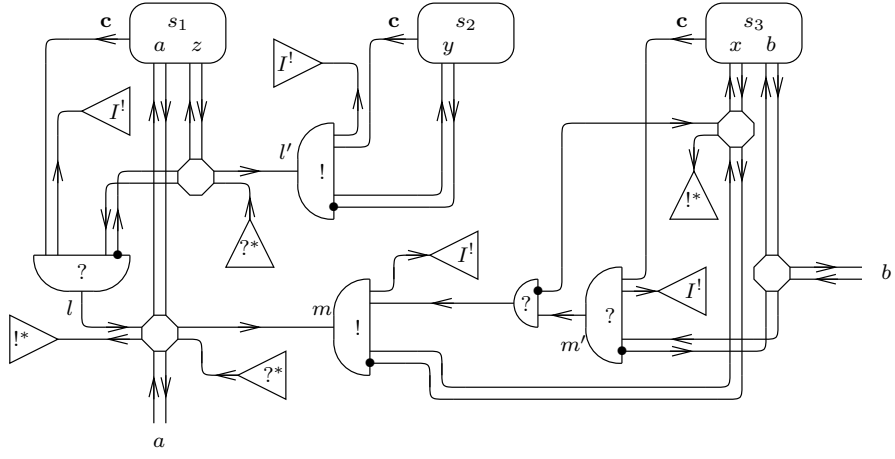


Fig. 28. Name passing: simple net s'_1 , translation of the process P' (after a few structural reductions)

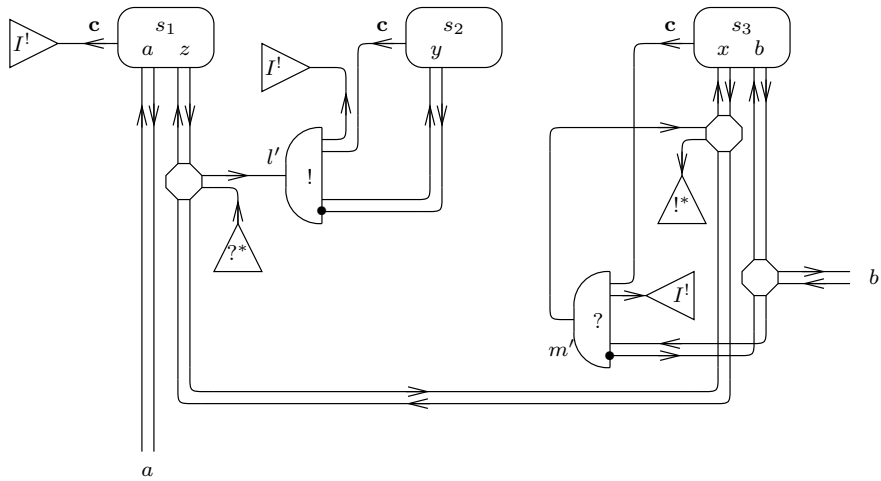


Fig. 29. Name passing: simple net s'_2 , first step

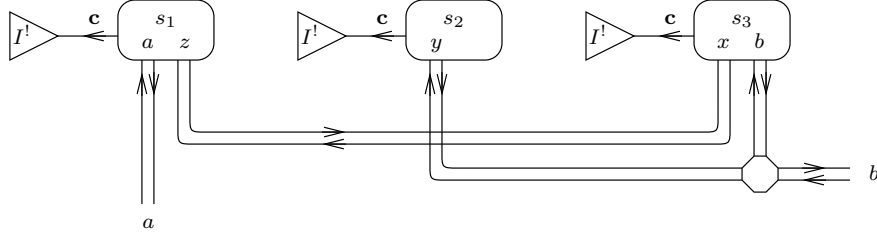


Fig. 30. Name passing: simple net s'_3 , final state

associated communication areas.

Finally $t \xrightarrow{l'\overline{m'}} t'$ with $t' \rightsquigarrow_d^* s'_3$ and s'_3 is the simple net of Figure 30 where y and b are also identified.

This corresponds to $(\Gamma, L) \rightsquigarrow_{\text{can}} ((\overline{l}a\langle z \rangle \cdot P, e[z \mapsto z'])([l']z(y) \cdot Q, e[z \mapsto z'])([m]a(x) \cdot \overline{m'}x\langle b \rangle \cdot R, e), \{z'\}) \xrightarrow{m\overline{l}} ((P, e[z \mapsto z'])([l']z(y) \cdot Q, e[z \mapsto z'])([m']x\langle b \rangle \cdot R, e[x \mapsto z']), \{z'\}) \xrightarrow{l'\overline{m'}} ((P, e[z \mapsto z'])(Q, e[z \mapsto z', y \mapsto b'])(R, e[x \mapsto z']), \{z'\})$ in the environment machine.

Conclusion. The main goal of this work was not to define one more translation of the π -calculus into yet another exotic formalism. We wanted to illustrate by our bisimulation result that differential interaction nets are sufficiently expressive for simulating concurrency and mobility, as formalized in the π -calculus. We believe that differential interaction nets have their own interest and find a strong mathematical and logical justification in their connection with linear logic, in the existence of various denotational models and in the analogy between its basic constructs and fundamental mathematical operations such as differentiation and convolution product. The fact that differential interaction nets support concurrency and mobility suggests that they might provide more convenient mathematical and logical foundations to concurrent computing. This work suggests that differential linear logic might be the logical side of a Curry-Howard correspondence for concurrency and mobility.

References

- [AC98] Roberto Amadio and Pierre-Louis Curien. *Domains and lambda-calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1998.
- [AM99] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1999.

- [Bef05] Emmanuel Beffara. *Logique, Réalisabilité et Concurrency*. PhD thesis, Université Denis Diderot, 2005.
- [BHY04] Martin Berger, Kohei Honda, and Nobuko Yoshida. Strong normalisability in the pi-calculus. *Information and Computation*, 191:145–202, 2004.
- [BM06] Emmanuel Beffara and François Maurel. Concurrent nets: a study of prefixing in process calculi. *Theoretical Computer Science*, 356(3):356–373, 2006.
- [CF06] Pierre-Louis Curien and Claudia Faggian. An approach to innocent strategies as graphs. Technical report, Preuves, Programmes et Systèmes, 2006. Submitted for publication.
- [Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005.
- [EL07] Thomas Ehrhard and Olivier Laurent. Acyclic solos. Technical report, Preuves, Programmes et Systèmes, 2007.
- [ER06] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006.
- [EW97] Uffe Engberg and Glynn Winskel. Completeness Results for Linear Logic on Petri Nets. *Annals of Pure and Applied Logic*, 86(2):101–135, 1997.
- [FM05] Claudia Faggian and François Maurel. Ludics nets, a game model of concurrent interaction. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, pages 376–385. IEEE Computer Society, 2005.
- [Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
- [Gir88a] Jean-Yves Girard. Geometry of interaction II: Deadlock free algorithm. In Martin-Löf & Mints, editor, *Proceedings of COLOG’88*, volume 417 of *Lecture Notes in Computer Science*, pages 76–93. Springer-Verlag, 1988.
- [Gir88b] Jean-Yves Girard. Normal functors, power series and the λ -calculus. *Annals of Pure and Applied Logic*, 37:129–177, 1988.
- [HL06] Kohei Honda and Olivier Laurent. Processes and polarized proof-nets. Technical report, Preuves, Programmes et Systèmes, 2006.
- [JM04] Ole Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical report, Cambridge University Computer Laboratory, 2004.
- [Laf95] Yves Lafont. From proof nets to interaction nets. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 225–247. Cambridge University Press, 1995. Proceedings of the Workshop on Linear Logic, Ithaca, New York, June 1993.

- [LPV01] Cosimo Laneve, Joachim Parrow, and Björn Victor. Solo diagrams. In *Proceedings of the 4th conference on Theoretical Aspects of Computer Science, TACS'01*, number 2215 in Lecture Notes in Computer Science. Springer-Verlag, 2001.
- [LV03] Cosimo Laneve and Björn Victor. Solos in concert. *Mathematical Structures in Computer Science*, 13(5):657–683, 2003.
- [Maz05] Damiano Mazza. Multiport interaction nets and concurrency. In *Proceedings of CONCUR 2005*, number 3653 in Lecture Notes in Computer Science, pages 21–35. Springer-Verlag, 2005.
- [Mel06] Paul-André Melliès. Asynchronous games 2: the true concurrency of innocence. *Theoretical Computer Science*, 358(2):200–228, 2006.
- [Mil93] Robin Milner. The polyadic pi-calculus: a tutorial. In *Logic and Algebra of Specification*, pages 203–246. Springer-Verlag, 1993.
- [Plo76] Gordon Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5(3):452–487, 1976.
- [Reg92] Laurent Regnier. *Lambda-Calcul et Réseaux*. Thèse de doctorat, Université Paris 7, January 1992.
- [SW01] Davide Sangiorgi and David Walker. *The pi-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.
- [Win87] Glynn Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer-Verlag, 1987.