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 replication/recursion into an untyped and essentially promotion-free 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.

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 towards "concurrent" interpretations of linear logic (e.g. [EW97], [AM99], [Mel06], [Bef05], [CF06] based on [FM05]...) missed a crucial feature of true concurrency, such as modelled by process calculi like Milner's π-calculus [Mil93,SW01]): its intrinsic *non-determinism*. This failure is easily understandable since there is an apparent contradiction between non-determinism and the Curry-Howard approach to computation consisting in identifying proofs and programs. According to this paradigm, a well-behaved proof system should possess a confluent cut-elimination procedure. But confluence is a way of expressing determinism in a rewriting setting: typically, it implies that a closed proof of boolean type cannot reduce to *true* and also to *false*.

Many denotational models of the lambda-calculus and of linear logic admit some form of non-determinisms (e.g. [Plo76,Gir88]), showing that a non-deterministic proof calculus is not necessarily trivial. The first author introduced such models, based on vector spaces in [Ehr02,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].

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. 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 [BHY03]. This translation can be considered as the first really convincing Curry-Howard interpretation of processes, but has two features which can be considered as slight defects: it accepts only replicable receivers and 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 nets[1].

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

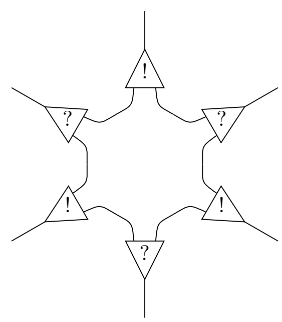


Fig. 1. Communication area

[1] 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 [BM05], 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.

Content. We first introduce differential interaction nets, typed with a recursive typing system (introduced by Danos and Regnier in [Reg92] and corresponding to the untyped lambda-calculus) for avoiding the appearance of non reducible configurations. This system is finitary in the sense that it has no promotion. Using these cells, 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 reduction. 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. Last we define a "translation" relation from machine states to nets and show that this translation relation is a bisimulation between the two transition systems.

The main goal of this work is not to define one more translation of the π-calculus into yet another exotic formalism. We want 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.

1 Differential interaction nets

1.1 Presentation of the cells

Our nets will be typed using a type system which corresponds to the untyped lambda-calculus. This typing system is based on a single type symbol o (the type of outputs), subject to the following recursive equation $o = ?o^{\perp} \parr o$. We set $\iota = o^{\perp}$, so that $\iota = !o \otimes \iota$ and $o = ?\iota \parr o$.

We assume known from the reader the basics of interaction nets, as introduced by Lafont in [Laf95], see also [ER06] for a more detailed introduction to differential interaction nets. In our pictures, cells are represented by triangles, and the principal port is located at one of the angles of the triangle. Sometimes, we shall put a black dot to locate the auxiliary port numbered 1. The other auxiliary ports are numbered in the obvious way, starting from this marked auxiliary port (the *arity* of the cell is the number of its auxiliary ports).

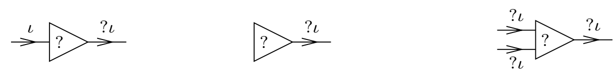
In the present setting, there are eleven kinds of cells: par (arity 2), bottom (arity 0), tensor (arity 2), one (arity 0), dereliction (arity 1), weakening (arity 0), contraction (arity 2), codereliction (arity 1), coweakening (arity 0), cocontraction (arity 2) and closed promotion (arity 0). We present now the various kinds of cells, with their typing rules, in a pictorial way.

1.1.1 Multiplicative cells.

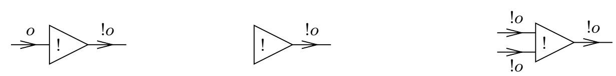
The *par* and *tensor* cells, as well as their "nullary" versions *bottom* and *one* are as follows:

1.1.2 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*:

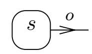
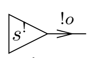


and then the *bang* cells, called *codereliction*, *coweakening* and *cocontraction*:



1.1.3 Closed promotion cells and simple nets.

The notion of simple net is then defined inductively, together with the notion of *closed promotion* cell.

Given a (non necessarily simple) net s with only one free port  we introduce a cell .

A *simple net* is a net, built according to the usual construction rules of typed interaction nets recorded in Section 6.1, using the kinds of cells we have introduced.

1.1.4 Nets.

A *net* is a finite 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.

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 and codereliction cells are equipped with labels belonging to $\mathcal{L}$. We require moreover that, if two labels occurring in a labeled net are equal, they are equal to τ. All the nets we consider in this paper are labeled. In our pictures, the labels of dereliction and codereliction cells will be indicated, unless it is τ, in which case the (co)dereliction cell will be drawn without any label.

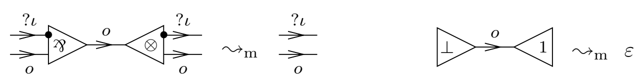
2 Reduction rules

We denote by Δ the collection of all simple nets and by $\mathbb{N}\langle\Delta\rangle$ the collection of all nets (finite sums of simple nets with the same interface).

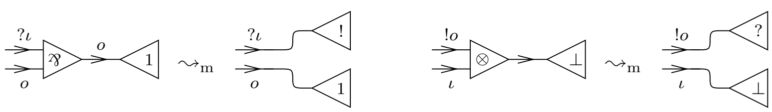
A *reduction rule* is a subset $\mathcal{R}$ of $\Delta \times \mathbb{N}\langle\Delta\rangle$ consisting of pairs (s, s') where s is made of two cells connected by their principal ports and s' has the same interface as s. This set can be finite or infinite. Such a relation is easily extended to arbitrary simple nets ($s \mathrel{\mathcal{R}} t$ if there is $(s_0, u_1 + \cdots + u_n) \in \mathcal{R}$ where s_0 is a subnet of s, each u_i is simple and $t = t_1 + \cdots + t_n$ where t_i is obtained by replacing s_0 by u_i in s). This relation is extended to nets (sums of simple nets): $s_1 + \cdots + s_n$ (where each s_i is simple) is related to s' by this extension $\mathcal{R}^\Sigma$ if $s' = s'_1 + \cdots + s'_n$ where, for each i, $s_i \mathrel{\mathcal{R}} s'_i$ or $s_i = s'_i$. Last, $\mathcal{R}^*$ is the transitive and reflexive closure of $\mathcal{R}^\Sigma$.

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.

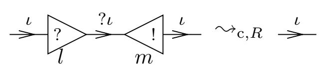


where ε stands for the empty simple net (not to be confused with the net $0 \in \mathbb{N}\langle \Delta \rangle$, which is not a simple net). The next two rules concern the interaction between a binary and a nullary multiplicative cell.



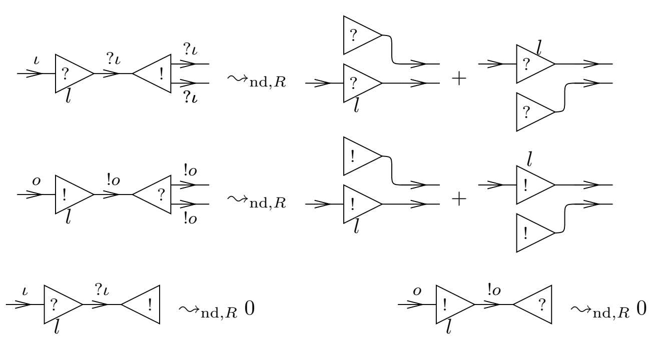
So here the reduction rule (denoted as $\rightsquigarrow_{\mathrm{m}}$) has four elements.

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

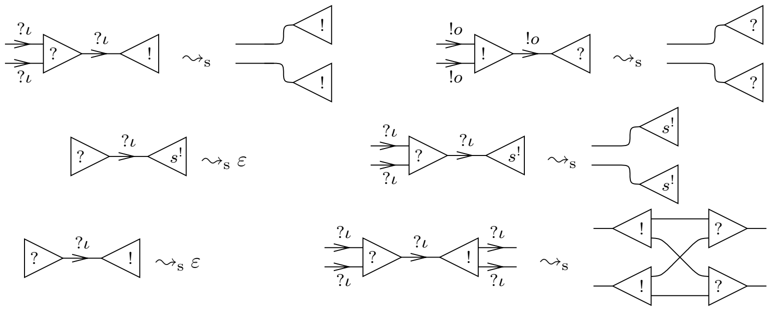


So the set $\rightsquigarrow_{\mathrm{c},R}$ is in bijective correspondence with the set of pairs (l, m) with $l, m \in R$ and $l = m \Rightarrow l = m = \tau$.

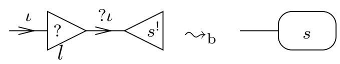
2.1.3 Non-deterministic reduction. Let $R \subseteq \mathcal{L}$. We have the following reductions if $l \in R$.



2.1.4 Structural reduction.



2.1.5 Box reduction.



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 promotion cell. Indeed, promotion is the only asymmetric rule of differential linear logic.

One can check that we have provided reduction rules for all possible redexes, compatible with our typing system: for any simple net[2] 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.

2.2 Confluence

Theorem 1. *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 $\rightsquigarrow_{\mathrm{c},R}$, $\rightsquigarrow_{\mathrm{nd},R'}$, $\rightsquigarrow_{\mathrm{m}}$, $\rightsquigarrow_{\mathrm{s}}$ and $\rightsquigarrow_{\mathrm{b}}$. 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: $\rightsquigarrow_R = \rightsquigarrow_{\mathrm{m}} \cup \rightsquigarrow_{\mathrm{c},\{\tau\}} \cup \rightsquigarrow_{\mathrm{s}} \cup \rightsquigarrow_{\mathrm{b}} \cup \rightsquigarrow_{\mathrm{nd},R}$. We set $\rightsquigarrow_{\mathrm{d}} = \rightsquigarrow_{\emptyset}$ ("d" for "deterministic") and denote by $\sim_{\mathrm{d}}$ 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 $\{l, m\}$**-neutrality.** Let l and m be distinct elements of $\mathcal{L} \setminus \{\tau\}$. We call (l, m)*-communication redex* a communication redex whose (co)dereliction cells

[2] And remember that such a structure must be typed.

are labeled by l and m. 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 1. *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.*

2.3.2 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 $s \xrightarrow{l\overline{m}} t$ if the following holds: $s \rightsquigarrow^*_{\{l,m\}} s_1 + s_2 + \cdots + s_n$ where s_1 is a simple net which contains an (l,m)-communication redex (with dereliction labeled by m and codereliction labeled by l) and becomes t when one reduces this redex, and each s_i (for $i > 1$) is $\{l,m\}$-neutral.

Lemma 2. *The relation $\sim_{\mathrm{d}} \subseteq \Delta \times \Delta$ is a a bisimulation on $\mathbb{D}_{\mathcal{L}}$.*

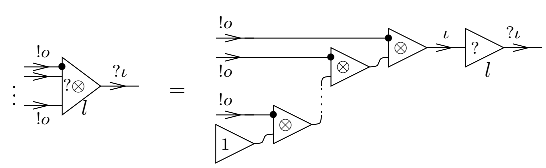
3 A toolbox for process calculi interpretation

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.

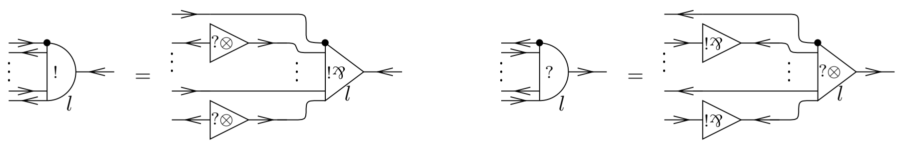
3.1.2 The dereliction-tensor and the codereliction-par cells. Let n be a non-negative integer. We define an n-ary cell as follows. It will be decorated by the label of its dereliction cell (if different from τ).



The number of tensor cells in this compound cell is equal to n. One defines dually the $!\invamp$ 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.

The *n-ary input cell* and the *n-ary output cell* are defined as



with n pairs of auxiliary ports.

Prefix cells are labeled by the label carried by their outermost dereliction-tensor or codereliction-par compound cell, if different from τ, the other codereliction-par or dereliction-tensor 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 solos[3].

These switches will be closed by "boxed identity cells", which are the unique use we make of promotion in the present work. Let I be the "identity" net of Figure 2.

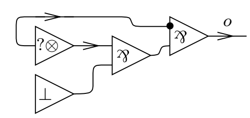
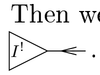


Fig. 2. Identity

Then we shall use the closed promotion cell labeled by $I^!$: .

3.2 Communication tools

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 3 shows how we picture a communication area of order 3.

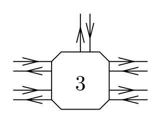
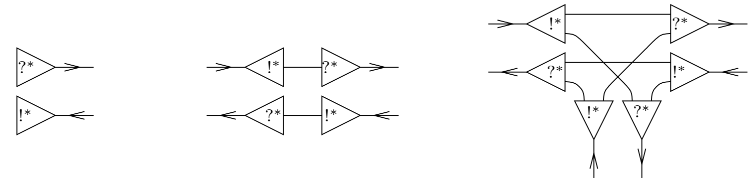


Fig. 3. Area of order 3

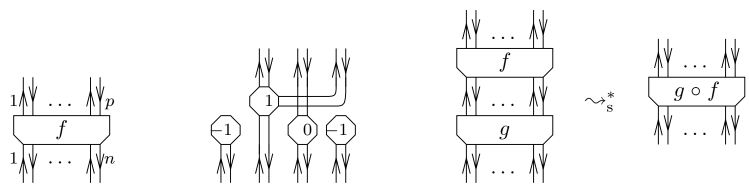
A communication area of order n is made of $n+2$ pairs of $(n+1)$-ary generalized cocontraction and contraction cells $(\gamma_1^+, \gamma_1^-), \ldots, (\gamma_{n+1}^+, \gamma_{n+1}^-)$, with, for each i and j such that $1 \leq i < j \leq n+2$, a wire from an auxiliary port of γ_i^+ to an auxiliary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

[3] 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 $P = \nu y\,(u(x,y) \mid y(\ldots)) \mid Q$, the first solo must interact before the second one with the environment Q.



3.2.2 Identification structures. Let $n, p \in \mathbb{N}$ and let $f : \{1, \ldots, p\} \to \{1, \ldots, n\}$ 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 4(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 three communication areas, two of order -1, one of order 0 and one of order 1, as in Figure 4(b).



(a) Notation (b) Example (c) Reduction

Fig. 4. Identification structures

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 and q, the resulting area is of order $p+q$, see Figure 5.

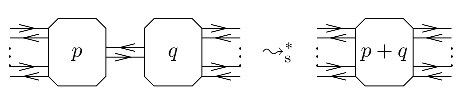
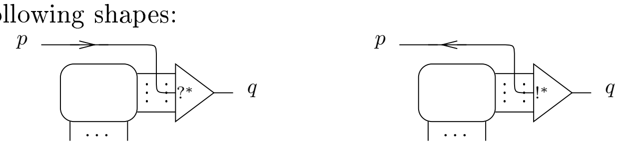


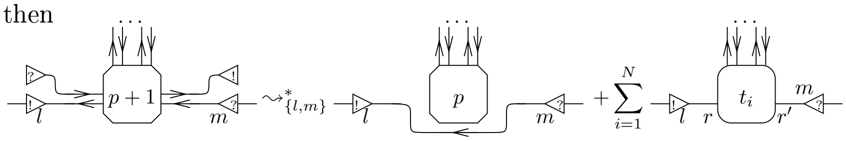
Fig. 5. Aggregation

3.3.2 Composition of identification structures. In particular, we get the reduction of Figure 4(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 following shapes:

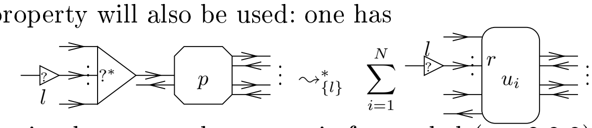


3.3.4 Forwarding of derelictions and coderelictions in communication areas. The following reduction shows that derelictions and coderelictions can meet eachother, when connected to a common communication areas. Let $l, m \in \mathcal{L}$, then



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.

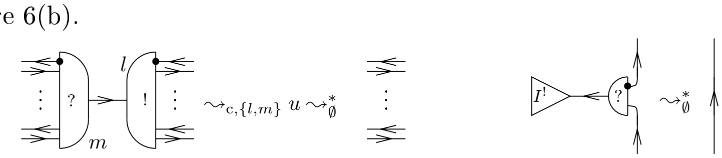
3.3.5 General forwarding. Let $l \in \mathcal{L}$. The following more general but less informative property will also be used: one has



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 $\rightsquigarrow_{c,\{l,m\}}$ to a net u which reduces by $\rightsquigarrow^*_{\{\tau\}}$ to 0 if $n \neq p$ and to simple wires, in Figure 6(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 6(b).



(a) Prefixes interaction

(b) Transistor triggering

Fig. 6. Prefix reduction

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 features. Let $\mathcal{N}$ be a countable set of names. Our processes are defined by the following syntax. We use the same set of labels as before.

- nil is the empty process.
- 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. a is bound in this process.
- If P is a process, $a, b_1, \ldots, b_n \in \mathcal{N}$, the names b_i being pairwise distinct and if $l \in \mathcal{L}$, then $Q = [l]a(b_1 \ldots b_n) \cdot P$ is a process (prefixed by an input action, whose subject is a and whose objects are the b_i; 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, $a, b_1, \ldots, b_n \in \mathcal{N}$ and $l \in \mathcal{L}$, then $\overline{[l]a}\langle b_1 \ldots b_n\rangle \cdot P$ 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 to be distinct. The name a can be equal to some of the b_is.

The purpose of this labeling of prefixes is to distinguish the various occurrences of names as subject of prefixes. The set $\mathsf{FV}(P)$ 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 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[4].

An *environment* is a function from a finite subset $\mathsf{Dom}\, e$ of $\mathcal{N}$ to a finite subset $\mathsf{Codom}\, e$ of $\mathcal{N}$. A *closure* is a pair (P, e) where P is a process and e is an environment such that $\mathsf{FV}(P) \subseteq \mathsf{Dom}(e)$. A *soup* is a multiset $S = (P_1, e_1) \cdots (P_N, e_N)$ of closures (denoted by simple juxtaposition). The codomain of a soup is the union of the codomains of the environments of this soup. The soup S is labeled if all the P_i's are labeled, with pairwise disjoint sets of labels. A *state* is a pair

[4] 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.

(S, L) where S is a soup and L is a set of names (the names which have to be considered as local to the state). The state (S, L) is labeled if the soup S is labeled.

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

4.1.1 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 $S = (P_1, e_1) \cdots (P_N, e_N)$ is *canonical* if each P_i is guarded, and that a state (S, L) is canonical if the soup S is canonical. One defines a rewriting relation $\rightsquigarrow_{\mathsf{can}}$ which allows to turn a state into a canonical one.

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

where, in the second rule, $a' \in \mathcal{N} \setminus (L \cup \mathsf{Codom}(e) \cup \mathsf{Codom}(S))$. One shows easily that, up to α-conversion, this reduction relation is confluent, and it is clearly strongly normalizing. We denote by $\mathsf{Can}(S, L)$ the normal form of the state (S, L) for this rewriting relation.

Moreover, observe that if $(S, L) \rightsquigarrow_{\mathsf{can}} (T, M)$, then (S, L) and (T, M) have the same set of free names.

4.1.2 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.

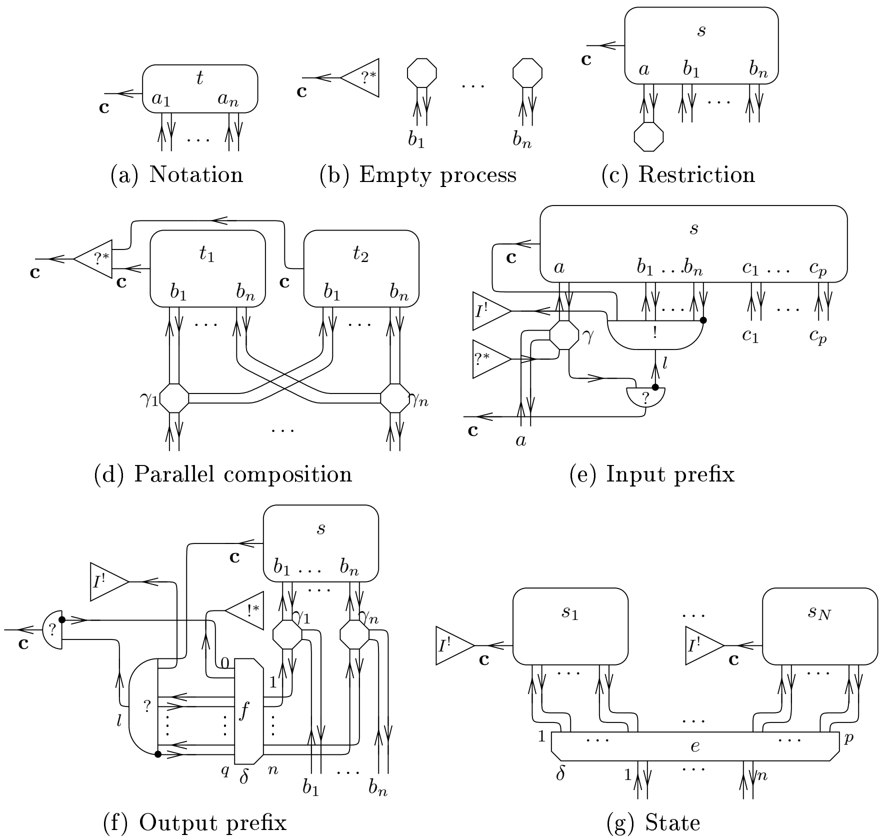
$$\begin{aligned}&((([l]a(b_1 \dots b_n) \cdot P, e)(\overline{[m]a'}\langle b'_1 \dots b'_n\rangle \cdot P', e')S, L)\\ &\quad \xrightarrow{l\overline{m}} \mathsf{Can}((P, e[b_1 \mapsto e'(b'_1), \dots, b_n \mapsto e'(b'_n)])(P', e')S, L)\end{aligned}$$

if $e(a) = e'(a')$. Observe that if $(S, L) \xrightarrow{l\overline{m}} (T, M)$ then $\mathsf{FV}(T, M) \subseteq \mathsf{FV}(S, L)$.

4.2 Translation of processes

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 nets t which have $2n + 1$ free ports $a_1^\iota, a_1^o, \dots, a_n^\iota, a_n^o$ and $\mathbf{c}$ as in Figure 7(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].

Clearly, if P and P' are α-equivalent, then $P \; \mathcal{I}_{a_1,\dots,a_n} \; s$ iff $P' \; \mathcal{I}_{a_1,\dots,a_n} \; s$.



(a) Notation (b) Empty process (c) Restriction (d) Parallel composition (e) Input prefix (f) Output prefix (g) State

Fig. 7. Process and state translation

4.2.1 Empty process. One has $\mathsf{nil}\ \mathcal{I}_{b_1,\ldots,b_n}\ t$ if t is t is as in Figure 7(b).

4.2.2 Name restriction. One has $\nu a \cdot P\ \mathcal{I}_{b_1,\ldots,b_n}\ t$ iff t is as in Figure 7(c), with s satisfying $P\ \mathcal{I}_{a,b_1,\ldots,b_n}\ s$.

4.2.3 Parallel composition. One has $P_1 \mid P_2\ \mathcal{I}_{b_1,\ldots,b_n}\ t$ iff the simple net t is as in Figure 7(d), where $P_1\ \mathcal{I}_{b_1,\ldots,b_n}\ t_1$, $P_2\ \mathcal{I}_{b_1,\ldots,b_n}\ t_2$ and $\gamma_1, \ldots, \gamma_n$ are communication areas of order 1.

4.2.4 Input prefix. Let $l \in \mathcal{L}$. Assume that $a, b_1, \ldots, b_n, c_1, \ldots, c_p$ are pairwise distinct names and let $Q = [l]a(b_1 \ldots b_n) \cdot P$. One has $Q\ \mathcal{I}_{a,c_1,\ldots,c_p}\ t$ if all the free names of P are contained in $a, b_1, \ldots, b_n, c_1, \ldots, c_p$ and if t is as in Figure 7(e), where γ is a communication area of order 1 and where s is a simple net which satisfies $P\ \mathcal{I}_{a,b_1,\ldots,b_n,c_1,\ldots,c_p}\ s$.

4.2.5 Output prefix. Let $l \in \mathcal{L}$. Let $b_1, \ldots, b_n$ be a list of pairwise distinct names and let $Q = \overline{[l]b_{f(0)}}\langle b_{f(1)} \ldots b_{f(q)}\rangle \cdot P$, where $f : \{0, 1, \ldots, q\} \to \{1, \ldots, n\}$ is a function. One has $Q\ \mathcal{I}_{b_1,\ldots,b_n}\ t$ if all the free names of P are contained in $b_1, \ldots, b_n$ and if t is as in Figure 7(f), where $\gamma_1, \ldots, \gamma_n$ are communication areas of order 1, δ is an f-identification structure and where s is a simple net which satisfies $P\,\mathcal{I}_{b_1,\ldots,b_n}\,s$.

4.2.6 States. Let $S = (P_1, e_1) \ldots (P_N, e_N)$ be a soup and $b_1, \ldots, b_n$ be a repetition-free list of names containing all the codomains of the environments $e_1, \ldots, e_N$. We assume that the domains of the environments e_i are pairwise disjoint, which is possible up to α-conversion. Let $a_1, \ldots, a_p$ be a repetition-free enumeration of the elements of $\bigcup_{i=1}^{N} \mathsf{Dom}\, e_i$, such that there is a list of non-negative integers $0 = h_0 \leq h_1 \leq \cdots \leq h_N = p$ such that, for $i = 1, \ldots, N$, the list $a_{h_{i-1}+1}, \ldots, a_{h_i}$ is a repetition-free enumeration of the elements of $\mathsf{Dom}(e_i)$. Let $e : \{1, \ldots, p\} \to \{1, \ldots, n\}$ be the map which is uniquely defined by the fact that, for each $i = 1, \ldots, N$ 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 $S\,\mathcal{I}_{b_1,\ldots,b_n}\,t$ if t is a simple net of the following shape, where $s_1, \ldots, s_N$ are simple nets such that $P_i\,\mathcal{I}_{b_1,\ldots,b_n}\,s_i$ and δ is an e-identification structure as in Figure 7(g).

Last, if we are moreover given $L \subseteq \mathcal{N}$ and a repetition-free list of names $b_1, \ldots, b_n$ containing all the free names of the state (S, L), one has $(S, L)\,\mathcal{I}_{b_1,\ldots,b_n}\,u$ if one has $S\,\mathcal{I}_{b_1,\ldots,b_n,c_1,\ldots,c_p}\,t$ for some repetition-free enumeration $c_1, \ldots, c_p$ of L (assumed of course to be disjoint from $b_1, \ldots, b_n$, 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_js.

5 Comparing the transition systems

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

Proposition 1. *Let (S, L) and (T, M) be canonical states and let $l, m \in \mathcal{L} \setminus \{\tau\}$. Assume that $(S, L) \xrightarrow{l\overline{m}} (T, M)$. Let s be a simple net such that $(S, L)\,\mathcal{I}_{b_1,\ldots,b_n}\,s$ where $b_1, \ldots, b_n$ is a repetition-free list of names containing all the free names of (S, L). Then there are simple nets t_0 and t such that $(T, M)\,\mathcal{I}_{b_1,\ldots,b_n}\,t$, $s \xrightarrow{l\overline{m}} t_0$ and $t_0 \sim_{\mathrm{d}} t$.*

Proposition 2. *Let (S, L) be a canonical state and $b_1, \ldots, b_n$ be a repetition-free list of names containing all the free names of (S, L). Let s be a simple net such that $(S, L)\,\mathcal{I}_{b_1,\ldots,b_n}\,s$. If t'_0 is a simple net such that $s \xrightarrow{l\overline{m}} t'_0$, then there is a canonical state (T, M) such that $(S, L) \xrightarrow{l\overline{m}} (T, M)$ and there exists a simple net t such that $(T, M)\,\mathcal{I}_{b_1,\ldots,b_n}\,t$ and $t \sim_{\mathrm{d}} t'_0$.*

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

Theorem 2. *The relation $\widetilde{\mathcal{I}}_{b_1,\ldots,b_n}$ defines a bisimulation between the labeled transition systems $\mathbb{S}_{\mathcal{L}}$ and $\mathbb{D}_{\mathcal{L}}$.*

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 Concurrence*. PhD thesis, Université Denis Diderot, 2005.

[BHY03] Martin Berger, Kohei Honda, and Nobuko Yoshida. Strong normalisability in the pi-calculus. *Information and Computation*, 2003. To appear.

[BM05] Emmanuel Beffara and François Maurel. Concurrent nets: a study of prefixing in process calculi. *Theoretical Computer Science*, 356, 2005.

[CF06] Pierre-Louis Curien and Claudia Faggian. An approach to innocent strategies as graphs. Technical report, Preuves, Programmes et Systèmes, 2006. Submitted for publication.

[Ehr02] Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12:579–623, 2002.

[Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005.

[ER06] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 2006. To appear.

[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.

[Gir88] 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. An exact correspondence between a typed pi-calculus and polarized proof-nets. In preparation, 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.

6 Annex: auxiliary notions

We recall here some notions which are not necessarily well known, and we also give some additional material about our particular way of presenting processes.

6.1 Reminder: the general formalism of interaction nets

Assume we are given a set of *kinds* and that an arity (a non-negative integer) and a typing rule is associated with each kind, this typing rule being a list $(A_0, A_1, \ldots, A_n)$ of types (where n is the arity associated to the kind; types are typically formulae of linear logic). A *net* is made of *cells*. With each cell γ is associated a kind and therefore an arity n and a typing rule $(A_0, A_1, \ldots, A_n)$. Such a cell γ has one *principal port* p_0 and n *auxiliary ports* $p_1, \ldots, p_n$. 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 $p_0, p_1, \ldots, p_n$ and typing rule is $(A_0, A_1, \ldots, A_n)$, denoting by $p'_0, p'_1, \ldots, p'_n$ the ports of the net uniquely defined by the fact that the sets $\{p_i, p'_i\}$ are wires (for $i = 0, 1, \ldots, n$), then the oriented wires (p_0, p'_0), (p'_1, p_1),$\ldots$,(p'_n, p_n) have type A_0, A_1,$\ldots$,A_n respectively.

6.2 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 $(\rho_1, \ldots, \rho_n)$ occurs as subject, it is always assumed that it has n objects $b_1, \ldots, b_n$, 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.

6.3 α-equivalence of states.

Given a partial function $f : \mathcal{N} \to \mathcal{N}$ and a process P, we denote by $f \cdot P$ the process where each free name a has been replaced by $f(a)$ (if $a \in \mathsf{Dom}\, f$) — 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 S and T are α-equivalent if $S = \gamma_1 \ldots \gamma_N$ and $T = \delta_1 \ldots \delta_N$ with $\gamma_i \sim_\alpha \delta_i$ for each i. Let $f : \mathcal{N} \to \mathcal{N}$ be a function. If $\gamma = (P, e)$ is a closure, one sets $f \cdot \gamma = (P, f \circ e)$. And last, $f \cdot (\gamma_1 \ldots \gamma_N) = (f \cdot \gamma_1) \ldots (f \cdot \gamma_N)$

Two states (S, L) and (T, 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 $f \cdot S \sim_\alpha T$. The free names of a state (S, L) are the names belonging to the codomain of S but not to L, we denote by $\mathsf{FV}(S, L)$ the set of these free names.

6.4 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

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

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:

$$\frac{}{[l]a(b_1, \ldots, b_n) \cdot P_1 \mid \overline{[m]}a\langle c_1, \ldots, c_n\rangle \cdot P_2 \xrightarrow{l\overline{m}} P_1[c_1, \ldots, c_n / b_1, \ldots, b_n] \mid P_2}$$

$$\frac{P_1 \sim P_1' \qquad P_1' \xrightarrow{l\overline{m}} P_2' \qquad P_2' \sim P_2}{P_1 \xrightarrow{l\overline{m}} P_2} \qquad \frac{P \xrightarrow{l\overline{m}} P'}{P \mid Q \xrightarrow{l\overline{m}} P' \mid Q}$$

$$\frac{P \xrightarrow{l\overline{m}} P'}{\nu a \cdot P \xrightarrow{l\overline{m}} \nu a \cdot P'}$$

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

Proposition 3. *For any process P, one has $P\ \mathcal{T}\ \mathsf{Can}((P, e), \emptyset)$ where e is the partial identity function whose domain is $\mathsf{FV}(P)$. Moreover, the relation $\mathcal{T}$ is a bisimulation.*

The proof is easy.

7 Annex: proofs

We give the proofs of all the statements of the paper.

7.1 Proof of Lemma 1

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 simple, contains an (l, m)-communication redex and one has actually $s_0 \rightsquigarrow^*_{\mathrm{d}} t_0$. Moreover, if s is the simple net obtained from s_0 by reducing the (l, m)-communication redex, then $s \rightsquigarrow_{\mathrm{d}} t$ where t is the simple net obtained from t_0 by reducing the (l, m)-communication redex of t_0.*

Now we can prove Lemma 1.

Proof. Assume $s \rightsquigarrow^*_{\{l,m\}} t = s_1 + \cdots + s_n$ 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'. □

7.2 Proof of Lemma 2

Proof. Let $s, s' \in \Delta$ and assume that $s \sim_{\mathrm{d}} s'$. Assume moreover that $s \xrightarrow{l\overline{m}} t$, which means that $s \rightsquigarrow^*_{\{l,m\}} s_0 + s_1 + \cdots + s_n$ 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 $\rightsquigarrow_{\mathrm{d}} \subseteq \rightsquigarrow^*_{\{l,m\}}$), there exists $u \in \mathbb{N}\langle\Delta\rangle$ such that $s_0 + s_1 + \cdots + s_n \rightsquigarrow^*_{\{l,m\}} u$ and $s' \rightsquigarrow^*_{\{l,m\}} u$. But by lemmas 3 and 1, we have $u = u_0 + u_1 + \cdots + u_m$ with $s_0 \rightsquigarrow_{\mathrm{d}} u_0$, u_0 contains an (l, m)-communication redex, and if we reduce this redex, we obtain a net t' such that $t \rightsquigarrow_{\mathrm{d}} t'$. □

7.3 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 lemmas which will be crucial in the proofs of Proposition 1 and 2.

7.3.1 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.

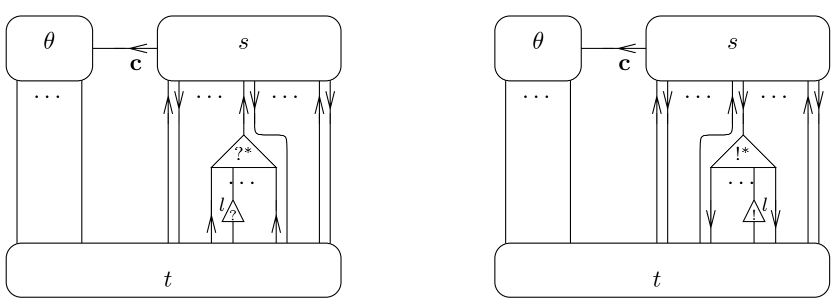
Such a sequence of ports will be called a *guarding path* from δ to r in s (observe that since $r \neq \tau$, there is no ambiguity on the (co)dereliction cell labeled by r in s, whereas l can be equal to τ and so there might be several (co)dereliction cells labeled by l in s).

7.3.2 Persistency.

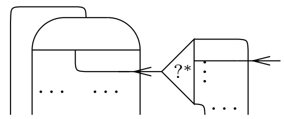
Lemma 4. *Let s be a simple net, let $R \subseteq \mathcal{L}$, let l, r be labels which are distinct, with $r \neq \tau$. Let δ be an l-labeled (co)dereliction cell which is guarded by r in s and assume that $s \rightsquigarrow_R^* s_1 + \cdots + s_p$ where the s_i are simple. Then δ and r occur, and 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). $\square$

7.3.3 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 following shapes (according to whether l labels a dereliction or a codereliction cell):



where θ is a boxed identity cell, or a net of the following shape, consisting of a labeled input of output prefix compound cell, with a label different from τ:



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

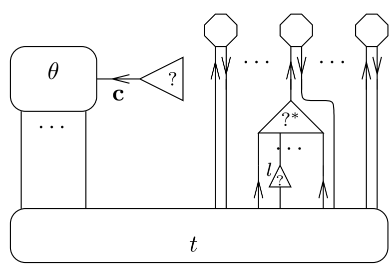
Lemma 5 (Diving). *Assume that $l \in \mathcal{L} \setminus \{\tau\}$ dives into P in the simple net u, and let $m \in \mathcal{L} \setminus \{\tau\}$ which does not occur in P. Then u is $\{l, m\}$-neutral.*

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 $\mathbf{c}$.

7.4 Proof of Lemma 5

Proof. By induction on P and contradiction, so assume that $u \rightsquigarrow^*_{\{l,m\}} u_1 + u'$ and that u_1 contains an (l, m)-communication redex.

Assume first that $P = \mathsf{nil}$. Assume that l is a dereliction. Then u has the following shape

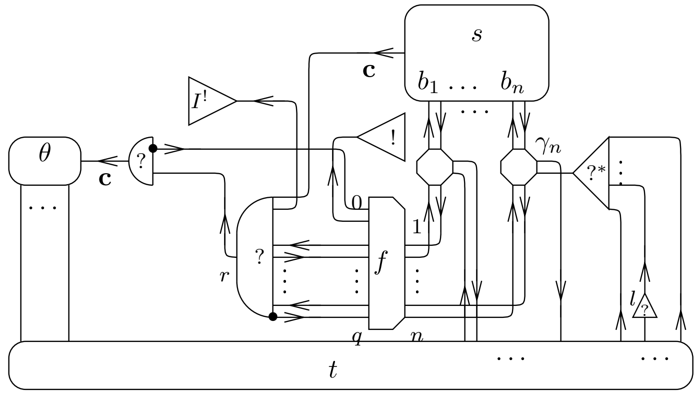


Thus $u \rightsquigarrow^*_{\{l,m\}} 0$ by 3.3.5. Hence by the Church-Rosser property of $\rightsquigarrow^*_{\{l,m\}}$, we must have $u_1 + u' \rightsquigarrow^*_{\{l,m\}} 0$. But this is impossible by Lemma 3 since u_1 has an (l, m)-communication redex.

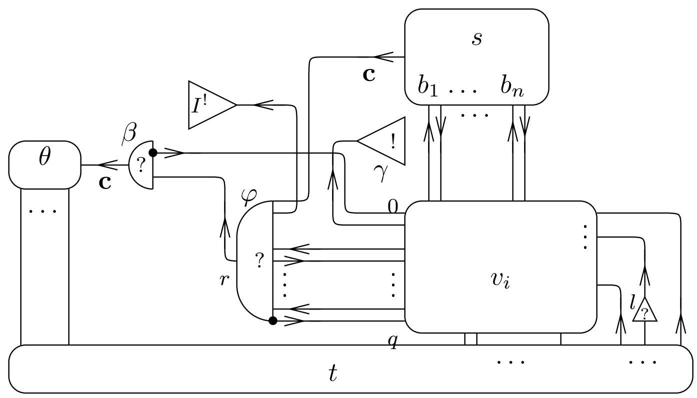
The case $P = P_1 \mid P_2$ is similarly handled: using 3.3.5 and the inductive hypothesis, one shows that $u \rightsquigarrow^*_{\{l,m\}} u'$ where u' is a sum of $\{l, m\}$-neutral simple nets, and hence u is $\{l, m\}$-neutral by Lemma 1.

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

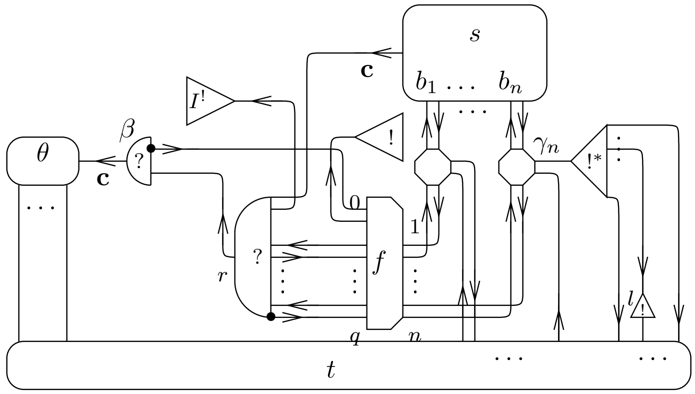


where, according to 3.3.5, in v_i, the principal port of l is forwarded (see the definition of this concept in 3.3.3)

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 4. For i satisfying (1), the inductive hypothesis applies, showing that u_i is $\{l,m\}$-neutral. Therefore u is $\{l,m\}$-neutral by Lemma 1.

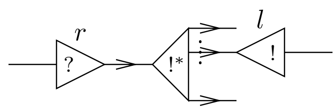
Assume now that l is a codereliction, so that u has the following shape (with the same notations as above).



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 $j \in \{1, \ldots, q\}$ 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 $\rightsquigarrow^*_{\{l,m\}}$-redex, and therefore, it will still be present in any simple summand of a net u'' such that $u' \rightsquigarrow^*_{\{l,m\}} u''$. So u' is $\{l, m\}$-neutral, and so is u by Lemma 1.

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 7.3.3). Here we can have $r' = m$, but l is guarded by r' in u, and hence u is $\{l, m\}$-neutral by Lemma 4 and Lemma 1.

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

Lemma 6. *Let (S, L) be a state and let $b_1, \ldots, b_n$ be a repetition-free enumeration of the free names of (S, L). Let (T, M) be its canonical form and let s be a simple net such that $(S, L)\ \mathcal{I}_{b_1,\ldots,b_n}\ s$. Then there exists a simple net t such that $(T, M)\ \mathcal{I}_{b_1,\ldots,b_n}\ t$ and $s \sim_{\mathrm{s}} t$.*

The proof is by simple inspection of the definition of the interpretation relation, using 3.3.1.

7.5 Proof of Proposition 1

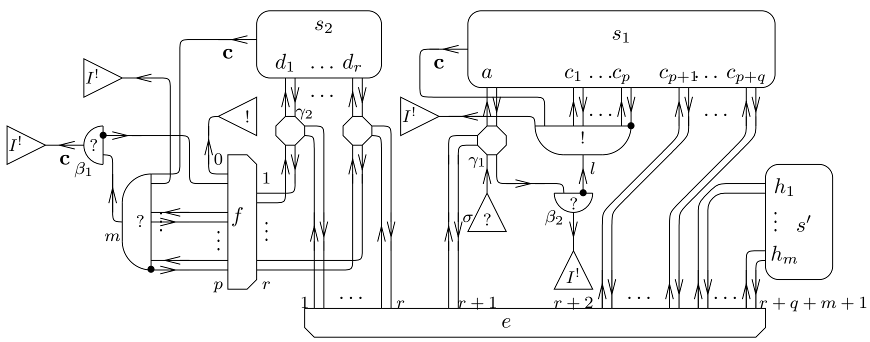
Proof. We know that S must be of the shape

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

where we assume that the e_i have pairwise disjoint domains, that a, $c_1, \ldots, c_p$, $c_{p+1} \ldots, c_{p+q}$ is a repetition-free enumeration of the domain of e_1, that $d_1, \ldots, d_r$ is a repetition-free enumeration of the domain of e_2, that $h_1, \ldots, h_m$ is a repetition-free enumeration of the union of the domains of $e_3, \ldots, e_N$, and $f : \{0, \ldots, p\} \to \{1, \ldots, r\}$ is a function, and we have $e_1(a) = e_2(d_{f(0)})$. And $(T, M) = \mathsf{Can}(S', L)$ where

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

Without loss of generality, we can assume that $f(0) = 1$. With these notations, s is the following simple net, where s_1 is a simple net such that $P\ \mathcal{I}_{a,c_1,\ldots,c_{p+q}}\ s_1$, s_2 is a simple net such that $Q\ \mathcal{I}_{d_1,\ldots,d_r}\ s_2$ and s' stands for the juxtaposition of simple nets s_i such that $P_i\ \mathcal{I}_{\boldsymbol{h^i}} s_i$ (for $3 \le i \le N$) where $\boldsymbol{h^i}$ stands for an enumeration of the domain of e_i (so that the lists of names $\boldsymbol{h^i}$ are pairwise disjoint, and their concatenation is a repetition-free enumeration of the names $h_1, \ldots, h_m$), with a boxed identity connected to the control ports of each s_i:



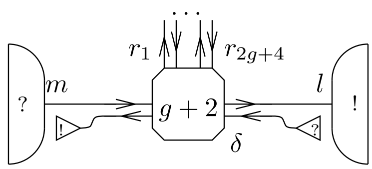
In this net, e is the function $\{1, \ldots, r{+}q{+}m{+}1\} \to \{1, \ldots, n\}$ which corresponds to the union of the functions e_i for $i = 1, \ldots, N$. Observe that we have $e(1) = e(r + 1)$ since by hypothesis $e_1(a) = e_2(d_1)$.

We have omitted in the picture the pairs of free ports corresponding to $b_1, \ldots, b_n, b_{n+1}, \ldots, b_{n+n'}$, 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 this net 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 γ_1, δ_1, γ_2 and δ_2 in an unique communication area δ. Let u be the obtained simple net, we have $s \rightsquigarrow^*_{\{l,m\}} u$.

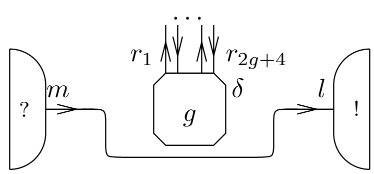
- Apply reduction 3.3.7 to both transistors β_1 and β_2 and let u' be the obtained simple net, we have $u \rightsquigarrow^*_{\{l,m\}} u'$.
- u' contains therefore the following subnet v



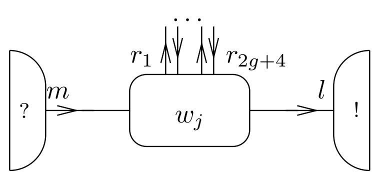
where, for $i = -1, 0, \ldots, g$ the pair of ports (r_{2i+3}, r_{2i+4}) is connected either

1. to the pair of port a of s_1
2. or to one of the pairs of ports $c_{p+1}, \ldots, c_{p+q}$ of s_1
3. or to one of the pairs of ports $h_1, \ldots, h_m$ of s'
4. or to a pair of ports of one of the communication areas connected to $d_2, \ldots, d_r$
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_i 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 $\rightsquigarrow^*_{\{l,m\}}$ reduction to a sum $v_0 + v_1 + \cdots + v_k$ where v_0 is

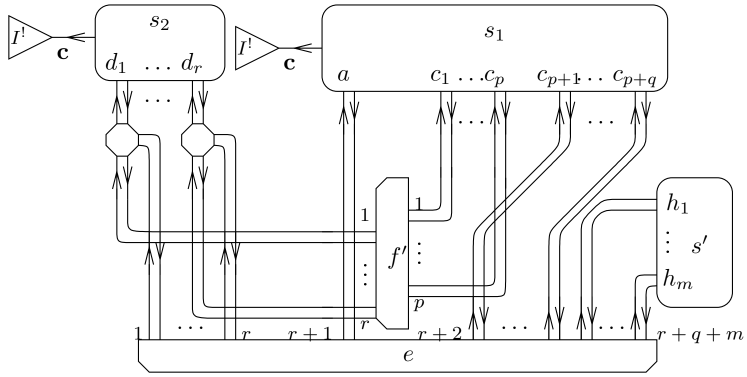


and the v_j's ($j \geq 1$) are nets of the shape



where the principal port of l and m are forwarded to ports among $r_1, \ldots, r_{2g+4}$. We have $u' \rightsquigarrow^*_{\{l,m\}} u'_0 + u'_1 + \cdots + u'_k$ where u'_j is obtained by replacing in u' the net v by the net v_j ($j = 0, \ldots, k$).

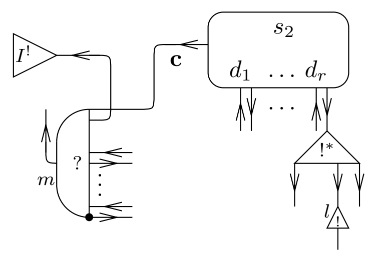
- We apply the (l, m)-communication reduction to u'_0, getting a simple net t_0 which is $\sim_{\mathrm{d}}$ equivalent to the following simple net



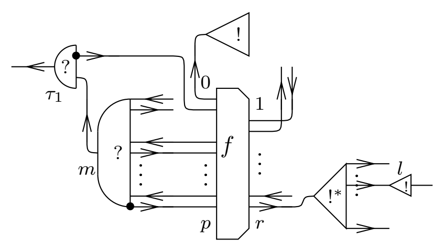
where f' is the restriction of f to $\{1, \ldots, p\}$. This net is $\sim_s$ equivalent to a simple net t_1 with $(S', L)\, \mathcal{I}_{b_1,\ldots,b_n}\, t_1$ (upon applying 3.3.1 to the communication areas of the identification structure f', the ones which are connected to the pairs of free ports d_i of s_2 and those belonging to the identification structure e). By Lemma 6, there is a simple net t such that $t_1 \sim_s t$ and $(T, M)\, \mathcal{I}_{b_1,\ldots,b_n}\, t$.

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 5.

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 shown shape (assuming that in u'_j, l is forwarded to the communication area connected to d_r). Hence by Lemma 5, w_1 is $\{l, m\}$-neutral.



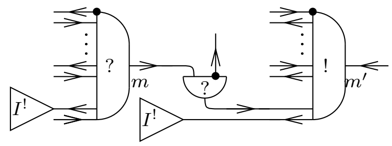
On the other hand, w_2 contains a subnet of the following shape. This subnet $\rightsquigarrow^*_{\{l,m\}}$ reduces by 3.3.5 to a sum of simple nets in each of which l is guarded by m. Therefore, by Lemma 1, w_2 is $\{l, m\}$-neutral. So by the same lemma, u'_j is $\{l, m\}$-neutral.



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 4. 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 $\square$

7.6 Proof of Proposition 2

Proof. One shows first that both l and m must be minimal in the poset $\mathcal{L}(S, L)$. 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 $\mathcal{L}(S, L)$). 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 $s \xrightarrow{l\overline{m}} t'_0$.

We have seen that l and m are minimal in the poset $\mathcal{L}(S, L)$ and this means that in S, the prefixes labeled by l and m are the outermost prefixes of P_1 and P_2 where $S = (P_1, e_1) \cdots (P_N, e_N)$ (and the choice of P_1 and P_2 is uniquely determined by l and m), that is, S is of the form described by Equation (1) in the proof of Proposition 1, 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 that formula, we argue now that necessarily $e_1(a) = e_2(d_{f(0)})$. But if this is not the case, an inspection of the interpretation of input prefixes (Paragraph 4.2.4), of states (Section 4.2.6) and of the identification structure (Paragraph 3.2.2) associated to the "global environment" e 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 $P_j \ \mathcal{I}_{c_1,\ldots,c_r} \ t$ for some $j = 1, \ldots, N$ and $c_1, \ldots, c_r$ 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 5. 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, Paragraph 4.2.5), 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 $Q \ \mathcal{I}_{h_1,\ldots,h_q} \ u$ (for a suitable list of names $h_1, \ldots, h_q$), 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 4 in the first case and Lemma 5 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 1. Finally, by the same lemma, s itself is $\{l,m\}$-neutral, contradicting the hypothesis that $s \xrightarrow{l\overline{m}} t'_0$.

So we must have $e_1(a) = e_2(d_{f(0)})$ and since our processes and states are implicitly arity-typed (see Paragraph 6.2), 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 1, and the statement itself of this theorem, we have $(S,L) \xrightarrow{l\overline{m}} (T,M)$ and there are simple nets t and t_0 such that $(T,M)\ \mathcal{I}_{b_1,\dots,b_n}\ t$, $t \sim_{\mathrm{d}} t_0$ and $s \xrightarrow{l\overline{m}} t_0$. This means more precisely that $s \rightsquigarrow^*_{\{l,m\}} s' = s_0 + s_1 + \cdots + s_p$, 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 $t_0 \sim_{\mathrm{d}} t'_0$.

We know from our hypothesis that $s \rightsquigarrow^*_{\{l,m\}} s'' = s'_0 + s'_1 + \cdots + s'_q$, 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 $s_0 \rightsquigarrow_{\mathrm{d}} u_0$ and $s'_0 \rightsquigarrow_{\mathrm{d}} u_0$, 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 $t_0 \rightsquigarrow_{\mathrm{d}} v_0$ and $t'_0 \rightsquigarrow_{\mathrm{d}} v_0$. So we have $t_0 \sim_{\mathrm{d}} t'_0$. □

7.7 Proof of Theorem 2

Proof. Let (S,L) be a canonical state and s_1 be a simple net, and assume that $(S,L)\ \widetilde{\mathcal{I}}_{b_1,\dots,b_n}\ s_1$. So there is a simple net s such that $(S,L)\ \mathcal{I}_{b_1,\dots,b_n}\ s$ and $s \sim_{\mathrm{d}} s_1$.

Assume first that $(S,L) \xrightarrow{l\overline{m}} (T,M)$, with l,m two distinct elements of $\mathcal{L}\backslash\{\tau\}$. By Proposition 1, there are simple nets t_0 and t such that $(T,M)\ \mathcal{I}_{b_1,\dots,b_n}\ t_0 \sim_{\mathrm{d}} t$ and $s \xrightarrow{l\overline{m}} t$. By Lemma 2 ($\sim_{\mathrm{d}}$ is a bisimulation), there exists t_1 such that $t \sim_{\mathrm{d}} t_1$ and $s_1 \xrightarrow{l\overline{m}} t_1$. We have $(T,M)\ \widetilde{\mathcal{I}}_{b_1,\dots,b_n}\ t_1$.

Conversely, assume that $s_1 \xrightarrow{l\overline{m}} t_1$. By Lemma 2, there exists t such that $t \sim_{\mathrm{d}} t_1$ and $s \xrightarrow{l\overline{m}} t$. By Proposition 2, there is a canonical state (T,M) and a simple net t_0 such that $(S,L) \xrightarrow{l\overline{m}} (T,M)$ and $(T,M)\ \mathcal{I}_{b_1,\dots,b_n}\ t_0 \sim_{\mathrm{d}} t$. We have $(T,M)\ \widetilde{\mathcal{I}}_{b_1,\dots,b_n}\ t_1$. □

8 Annex: examples

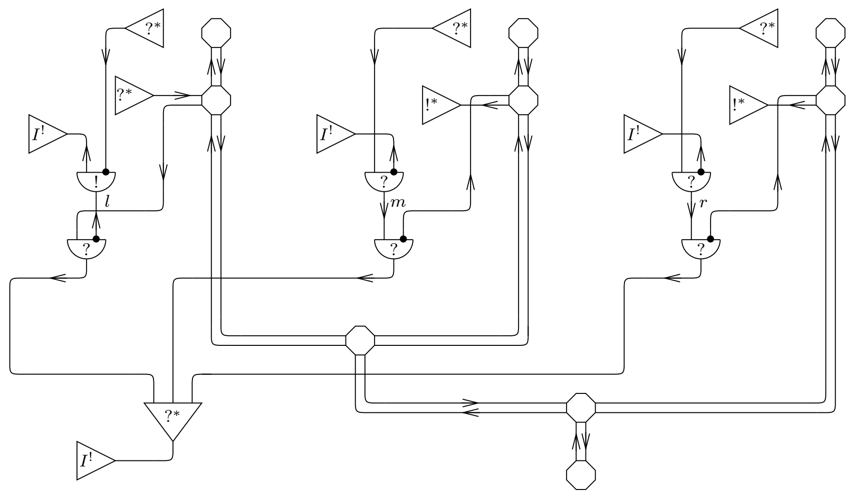
We give a few examples to illustrate some key features of communication in the π-calculus as represented in differential interaction nets.

8.1 Concurrent communication

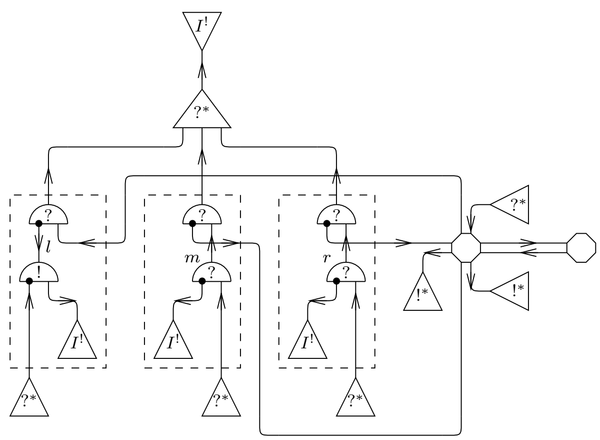
Let P be the process:

$$\nu a \cdot \Big(\Big([l]a() \cdot \mathsf{nil} \mid \overline{[m]a}\langle\rangle \cdot \mathsf{nil}\Big) \mid \overline{[r]a}\langle\rangle \cdot \mathsf{nil}\Big)$$

The simplest state containing P is $(S, L) = ((P, \emptyset), \emptyset)$. We have $(S, L)\ \mathcal{I}\ s$ where s is the following simple net:

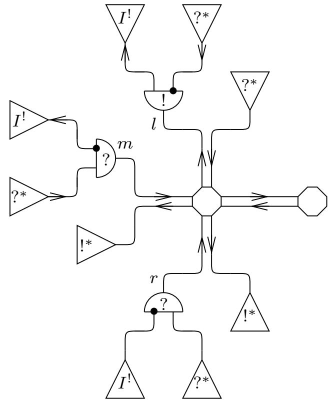


By applying aggregations of communication areas, we obtain the simple net s_1:

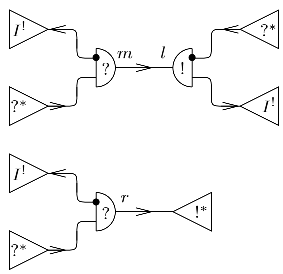


thus $s \rightsquigarrow_{\mathrm{s}}^{*} s_1$. Since P is in fact a CCS process, we can remark how the translation into differential interaction nets is given by first a tree (with nodes represented with dashed boxes) corresponding to the tree structure of the CCS process (built from sequential and parallel compositions), and second communication areas for the identification of names.

The simple net s_1 reduces to the following net s_2 ($s_1 \rightsquigarrow_{\mathrm{d}}^{*} s_2$):



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 following s_3 ($s_2 \rightsquigarrow^*_{\{l,m\}} s_3 + \cdots$):



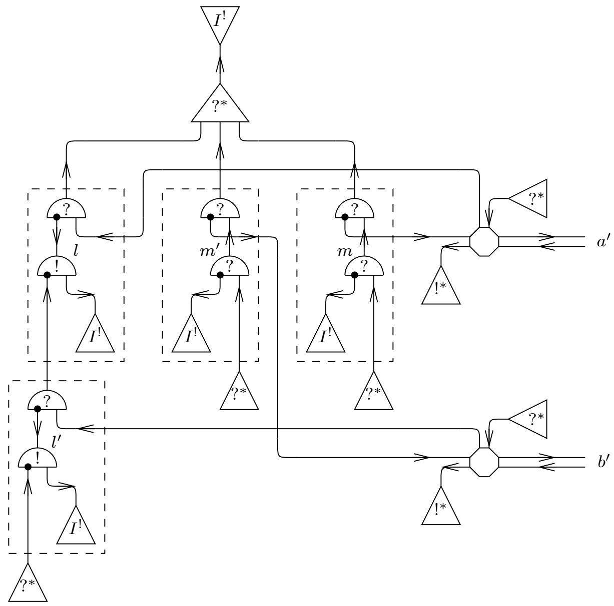
If t is obtained from s_3 by reducing the (l, m)-communication redex, we have $s \xrightarrow{l\overline{m}} t$. This corresponds to $(S, L) \rightsquigarrow_{\mathsf{can}} ((([l]a() \cdot \mathsf{nil}, e)(\overline{[m]a}\langle\rangle \cdot \mathsf{nil}, e)(\overline{[r]a}\langle\rangle \cdot \mathsf{nil}, e), \{a'\}) \xrightarrow{l\overline{m}} ((\overline{[r]a}\langle\rangle \cdot \mathsf{nil}, e), \{a'\})$ (with e defined only on $\{a\}$ by $e(a) = a'$) in the environment machine.

8.2 Sequentiality

Let P be the process:

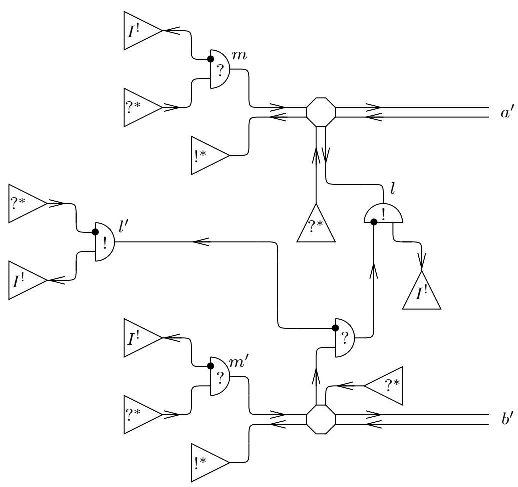
$$\nu a \cdot \Big([l]a() \cdot [l']b() \cdot \mathsf{nil} \mid \overline{[m']b}\langle\rangle \cdot \mathsf{nil} \mid \overline{[m]a}\langle\rangle \cdot \mathsf{nil}\Big)$$

The simplest state containing P is $(S, L) = ((P, e), \emptyset)$ (with e defined on $\{a, b\}$ by $e(a) = a'$ and $e(b) = b'$). We have $(S, L)\ \mathcal{I}_{a',b'}\ s$ with $s \rightsquigarrow^*_{\mathsf{s}} s_1$ (aggregations of communication areas) and s_1 is the following simple net:

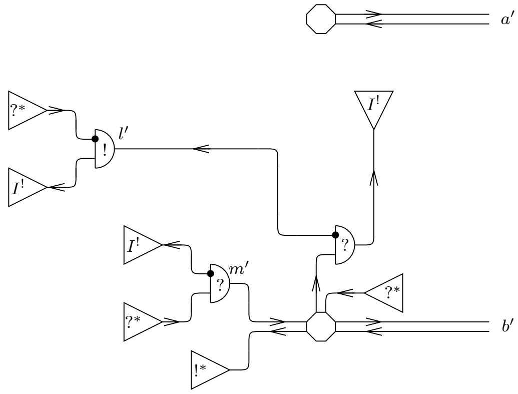


Since P is again a CCS process, we can see its tree structure in the differential interaction net s_1.

The simple net s_1 reduces to the following net s_2 ($s_1 \rightsquigarrow_{\mathrm{d}}^* s_2$):



Then there exists a simple net s_3 such that $s_2 \rightsquigarrow_{\{l,m\}}^* s_3 + \cdots$ and if t is obtained from s_3 by reducing the (l,m)-communication redex it contains, we have $s \xrightarrow{l\overline{m}} t$. Moreover t reduces to the following net:



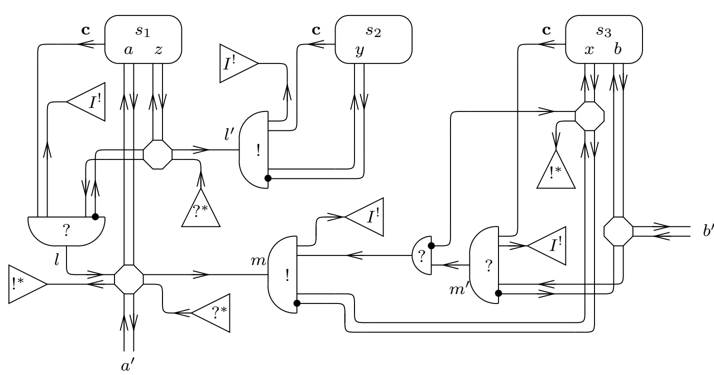
This corresponds to $(S,L) \rightsquigarrow_{\mathsf{can}} (([l]a() \cdot [l']b() \cdot \mathsf{nil}, e)(\overline{[m']b}\langle\rangle \cdot \mathsf{nil}, e)(\overline{[m]a}\langle\rangle \cdot \mathsf{nil}, e), \emptyset) \xrightarrow{l\overline{m}} (([l']b() \cdot \mathsf{nil}, e)(\overline{[m']b}\langle\rangle \cdot \mathsf{nil}, e), \emptyset)$ in the environment machine.

8.3 Name passing

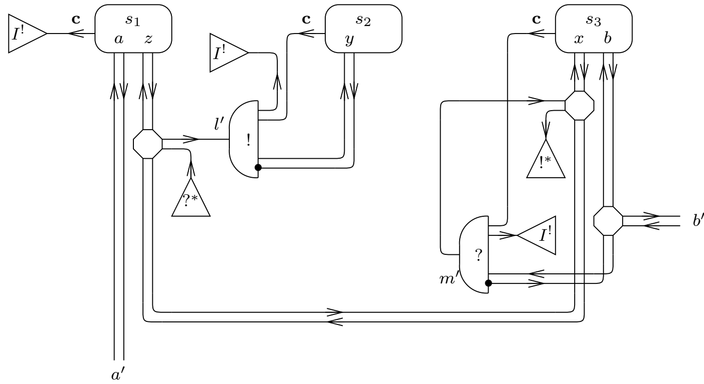
Let P, Q and R be processes such that the free names of P are a and z, the only free name of Q is y and the free names of R are x and b. Let P' be the process:

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

The simplest state containing P' is $(S,L) = ((P',e),\emptyset)$ (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 $(S,L)\ \mathcal{I}_{a',b'}\ s'$ with $s' \rightsquigarrow_{\mathrm{s}}^* s_1'$ (aggregations of communication areas) and s_1' is the following simple net:

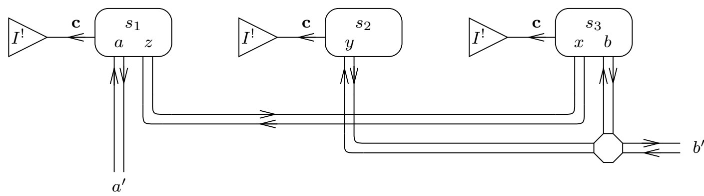


We have $s' \xrightarrow{m\bar{l}} t$ with $t \rightsquigarrow_{\mathrm{d}}^* s_2'$ and s_2' is the following simple net:



where the identification of the names z and x corresponds to the connection of the associated communication areas.

Finally $t \xrightarrow{l'\overline{m'}} t'$ with $t' \rightsquigarrow_{\mathrm{d}}^{*} s_3'$ and s_3' is the following simple net:



where y and b are also identified.

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