Interaction Combinators

Yves Lafont*

Institut de Mathématiques de Luminy, UPR 9016 du CNRS,
163 avenue de Luminy, case 930, 13288 MARSEILLE CEDEX 9, France
Email: lafont@iml.univ-mrs.fr

March 25, 1997

It is shown that a very simple system of *interaction combinators*, with only 3 symbols and 6 rules, is a universal model of distributed computation, in a sense that will be made precise. This paper is the continuation of the author's work on *interaction nets*, inspired by Girard's proof nets for *linear logic*, but no preliminary knowledge of these topics is required for its reading.

Introduction

This paper addresses the following question: what are the fundamental laws of computation? Of course, the answer depends on the choice of a particular model of computation. Let us mention some of them:

- the *Turing machines* imitate the mathematician writing symbols on paper. There are many variants, for instance the *register machines* and the *stack machines*. This notion has the advantage of being simple and powerful at the same time, but it models only sequential computation;

- the *cellular automata* can be seen as discrete approximations of physical processes. This notion models distributed computation, but with a global synchronization of transitions;

- the *rewrite systems* are closer to the algebraic tradition, since a rewrite rule is just an oriented equation. An interesting example is the λ-*calculus*, with only one rewrite rule:
$$(\lambda x.u)v \to v[u/x].$$
This calculus is *Turing complete* and it has a nice logical interpretation, at least in the typed case. However, the rule is more complicated than it seems: is it reasonable to consider substitution as an atomic operation? In this sense, a more primitive system is the *combinatory logic*, with two rewrite rules:
$$\mathbf{K}xy \to x, \qquad \mathbf{S}xyz \to xz(yz).$$
But again, is it reasonable to consider erasing and duplication as atomic operations?

Following this tradition of rewrite systems, the *interaction nets* have been introduced in [Laf90] as a model of distributed computation with local synchronization (section 1). These nets, which are related to the *connection graphs* of [Baw86], appeared as a generalization of Girard's proof nets for linear logic (see [Gir95, Laf95]).

By "local synchronization", we mean that there is no need to consider a global time for computation. In other words, times is relativistic. By "distributed", we mean that the computation is performed at several places at the same time, whereas "parallel" sometimes refers to a kind of magical superposition, as in "parallel or":
$$\mathbf{T} \vee x \to \mathbf{T}, \qquad x \vee \mathbf{T} \to \mathbf{T}, \qquad \mathbf{F} \vee \mathbf{F} \to \mathbf{F}.$$
Our interaction nets are deterministic in a strong sense: not only the result, but also the computation is unique, up to trivial commutations. In particular, it is not possible to encode "parallel or". We shall not address the question of deciding whether this should be considered as a good or a bad point.

From the viewpoint of computability, our interaction nets are equivalent to the Turing machines, but from the viewpoint of computation, there is something more, for instance parallelism (in the sense of distributed computation). To express this rigorously, we introduce a natural notion of *translation of interaction system* preserving the essential properties of computations, such as the complexity and the degree of parallelism.

By definition, a *universal interaction system* has the property that any other interaction system can be translated into it. Turing machines can be seen as particular interaction systems, but such systems are intrinsically sequential and cannot be universal in the above sense, even if

*Partially supported by HCM Project CHRX-CT93-0046 (*Lambda Calcul Typé*).

they come from universal Turing machines. On the other hand, it is proved that a system of *interaction combinators* is universal (section 2). This suggests an answer to our original question, at least within the framework of interaction nets: the fundamental laws of computation are *commutation* and *annihilation*.

Our system of interaction combinators has been obtained by a kind of distillation, starting from some more complicated system suggested by Samson Abramsky for implementing the proof boxes of linear logic (see [Mac94]). Independently, Simon Gay has also obtained a universal system, with 8 symbols instead of 3 (see [Gay95]). Our system is simpler because it uses the same symbols for different purposes. Among the other related systems, let us mention the infinite one introduced by John Lamping for the optimal reduction of λ-calculus (see [Lam90]) and the variants proposed by Georges Gonthier, Martin Abadi, and Jean-Jacques Levy, in connection with linear logic (see [GAL92a, GAL92b]).

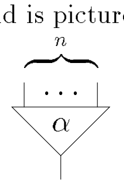
Apart from its simplicity, the system of interaction combinators has an unexpected interpretation in terms of reversible 2-stack machines which seems to throw a bridge between distributed and sequential computation (section 3). This section benefited from discussions with Vincent Danos and Laurent Regnier.

1 Interaction nets

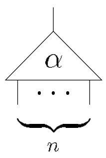
The origin of our favorite model of computation is explained in [Laf95]. Here, it is introduced from scratch, without explicit reference to proof theory.

1.1 Nets

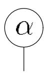
From now on, a symbol α will always be given with its *arity* $n \geq 0$. An occurrence of such a symbol is called a *cell*, and is pictured as follows:



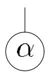
or



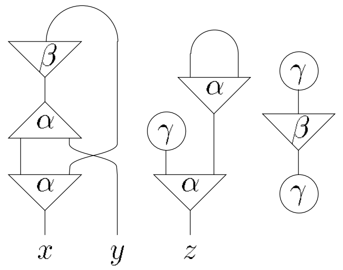
Such a cell has one *principal port* and n *auxiliary ports*. It is well understood that the latter are not interchangeable. For instance, one can number them from 1 to n, keeping 0 for the principal port. In practice, the ports will always be implicitly numbered in clockwise order. In particular, the second picture is obtained from the first one by a rotation of 180 degrees. If $n = 0$, there is no auxiliary port, and the cell is pictured as follows:



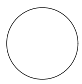
or



A *net* is a graph consisting of a finite number of cells and an extra set of *free ports*, each port being connected to another one by means of a *wire*. For instance, here is a net built with the symbols α, β and γ, of respective arities 2, 1 and 0:



This net has 3 free ports x, y and z. Note that a net is not necessarily connected, and that a wire may connect two ports of the same cell. In fact, we shall also allow *cyclic wires* which do not connect any ports:

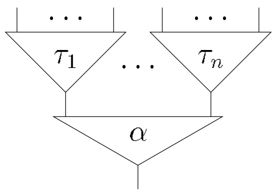


To sum up, a net is completely described by the following data:

- a finite set X (of free ports),
- a finite set C (of cells),
- a symbol $\ell(c)$ for each $c \in C$,
- a finite set W (of wires),
- a set ∂w of 0 or 2 *ports* for each $w \in W$.

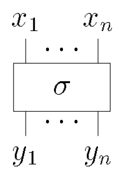
Here, a *port* is either an element of X, or a pair (c, i) where $c \in C$ and i ranges from 0 to the arity of $\ell(c)$. The nonempty ∂w must define a partition of all ports. In this paper, we shall avoid such cumbersome descriptions, but the reader may check that our arguments can always be formalized in this way.

If α is a symbol of arity n, an α-cell can be seen as a net with $n+1$ free ports. More generally, a *tree* is a net τ with one distinguished free port, called the *root*. It is either a single wire, in which case the root is fixed arbitrarily, or it is obtained by plugging the n auxiliary ports of a cell into the roots of smaller trees $\tau_1, \ldots, \tau_n$:

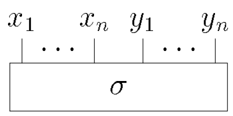


In that case, the root is the free port which is connected to the principal port of this cell.

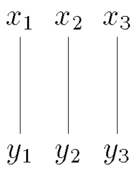
A *wiring* is a net ω without cell and without cyclic wire. So it is just a pairing of its free ports. In particular, a wiring has an even number of free ports. A permutation σ of $\{1, \ldots, n\}$ defines a wiring with $2n$ free ports, which is represented as follows:



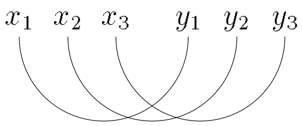
or



In both cases, x_i is connected to $y_{\sigma(i)}$. For instance, the wiring corresponding to the identity on $\{1, 2, 3\}$ is pictured as follows:

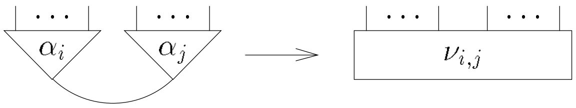


or

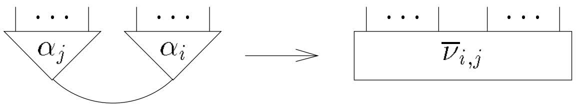


1.2 Interaction

Fix a finite alphabet Σ with m symbols $\alpha_1, \ldots, \alpha_m$ of respective arities $n_1, \ldots, n_m$. An *interaction rule* is a reduction of the form



where $\nu_{i,j}$ is a net with $n_i + n_j$ free ports. Note that cells can only interact pairwise, through their principal ports. Cyclic wires may be created when such a rule is applied inside a net, and this is why they have been allowed in the definition of nets. The above rule is clearly equivalent to



where $\overline{\nu}_{i,j}$ is obtained by exchanging the n_i first free ports with the n_j last ones in $\nu_{i,j}$.

An *interaction system* is a set of interaction rules which can be applied without any ambiguity. More precisely:

- if a rule is given for α_i, α_j, then no other one is given for α_i, α_j, or for α_j, α_i;
- if a symbol α_i interacts with itself[1] then $\nu_{i,i} = \overline{\nu}_{i,i}$.

[1] In [Laf90], symbols were not allowed to interact with themselves, but it appeared later that this was too restrictive.

In particular, an interaction system is necessarily finite, with at most $m(m+1)/2$ rules. Alternatively, such a system can be described by a partially defined square matrix $(\nu_{i,j})_{1 \le i,j \le m}$ satisfying the following *symmetry condition*:

- if $\nu_{i,j}$ is defined, so is $\nu_{j,i}$, and $\nu_{j,i} = \overline{\nu}_{i,j}$.

Later we shall add the technical condition that the $\nu_{i,j}$ are *reduced nets.*

Proposition 1 *If a net μ reduces in one step to ν and to ν', with $\nu \neq \nu'$, then ν and ν' reduce in one step to a common net ξ.*

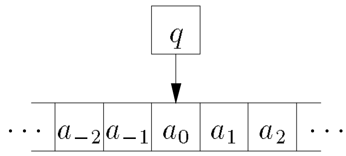
$$\begin{tikzcd} & \mu \arrow[dl] \arrow[dr] & \\ \nu \arrow[dr, dashed] & & \nu' \arrow[dl, dashed] \\ & \xi & \end{tikzcd}$$

Proof: In an instance of the left member of a rule, the cells are connected through their principal ports, so that two such instances are necessarily disjoint and the corresponding rules can be applied independently. **Q.e.d.**

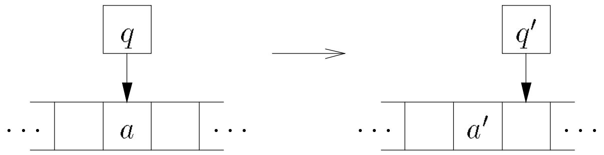
This strong confluence property has the following consequence: if a net μ reduces to an irreducible net ν in n steps, then any reduction starting from μ eventually reaches ν in n steps. Moreover, if one abstracts from the irrelevant order of application of rules, there is only one possible reduction from μ to ν. So we can say that interaction nets are a deterministic and asynchronous model of computation. In fact, we think that any computation of that kind can be modeled by means of interaction nets, but of course, an assertion of this kind cannot be proved.

1.3 Example: Turing machines

Classical models of sequential computation such as Turing machines, register machines and stack machines can be seen as special classes of interaction systems. For instance, a Turing machine is given by a triple (Q, A, T), where Q is a finite set of states, A a finite set of letters, and T a (partially defined) map from $Q \times A$ to $A \times \{+, -\} \times Q$. A configuration of the machine is given by a state q, an infinite tape filled with letters, and a current position in the tape:

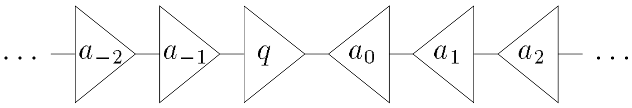


If $T(q, a) = (a', +, q')$, and being in state q, the machine reads a from the tape, then it writes a', moves right, and goes to state q':

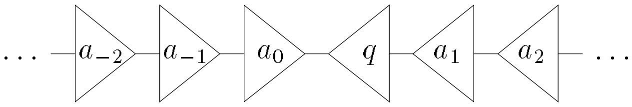


If $T(q,a) = (a', -, q')$, the machine has the same behavior, except that it moves left. By doubling the number of states, one can always assume that the direction of the move depends only on q'. One says that q' *comes from the left* in the first case, and that it *comes from the right* in the second case.

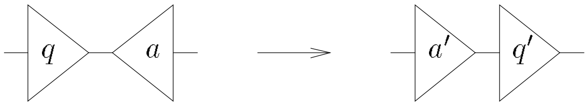
All states and letters are now considered as symbols of arity 1. The above configuration is simulated by the net



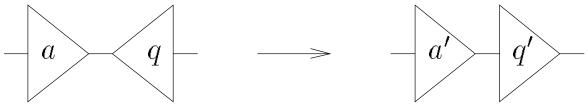
or



depending on whether q comes from the left or from the right. Similarly, the above transition is simulated by the rule



or

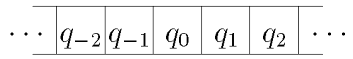


depending on whether q comes from the left or from the right. Of course, all this assumes that we have an infinite net! To simulate computations with a finite net, one must add a new symbol of arity 0 corresponding to the boundary of (the written part of) the tape, and some appropriate rules to cope with this new symbol. The details are left to the reader.

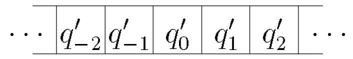
So it appears that interaction nets are complete from the viewpoint of computability. Similarly, the register machines and the stack machines can be simulated in a fairly obvious way.

1.4 Example: cellular automata

More surprisingly, the cellular automata, which are a synchronous model of distributed computation, can also be simulated by means of interaction nets. For instance, a 1-dimensional cellular automaton is given by a pair (Q, T), where Q is a finite set of states and T is a map from $Q \times Q \times Q$ to Q. A configuration of the automaton is given by an infinite sequence of states:

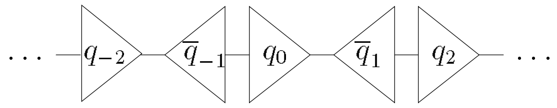


After one step of computation, it becomes

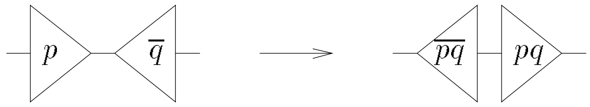


where $q'_i = T(q_{i-1}, q_i, q_{i+1})$. In particular, this means that all cells change at the same time.

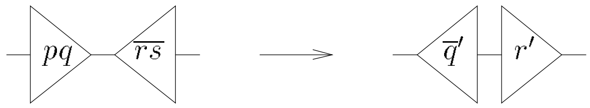
Two symbols q and $\overline{q}$ are introduced for each state q, and also two symbols pq and $\overline{pq}$ for each pair p, q. The above configuration is simulated by



and the transitions by



and



where $q' = T(p,q,r)$ and $r' = T(q,r,s)$. This simply means that a transition is decomposed into two steps: each cell interacts first with its left (or right) neighbor, and then with the other one. In this way, the global synchronization is no more needed.

The main difference between the system for Turing machines and the one for cellular automata appears in the structure of the right members of rules: in the first case, only one free port is connected to a principal port, whereas in the second case, both free ports are connected to principal ports. In fact, a system of the first kind can only model sequential computation.

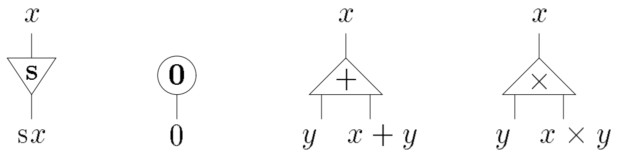
1.5 Example: unary arithmetics

Now we shall build a simple interaction system for unary arithmetics, using the following symbols: **s** (*successor*) of arity 1, **0** of arity 0, $+$ and $\times$, both of arity 2. The first two symbols are used to represent natural numbers in unary form. One starts from the usual equations defining addition and multiplication:

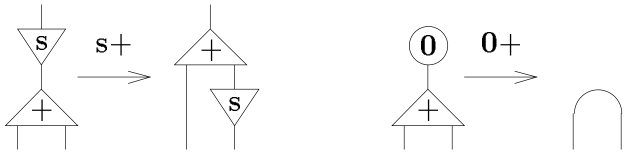
$$\mathrm{s}x + y = \mathrm{s}(x+y), \qquad 0 + y = y,$$

$$\mathrm{s}x \times y = (x \times y) + y, \qquad 0 \times y = 0.$$

Since addition is defined by induction on the first argument, we shall always plug this argument into the principal port of $+$, and similarly for $\times$. Therefore, the ports will be interpreted in the following unusual way:

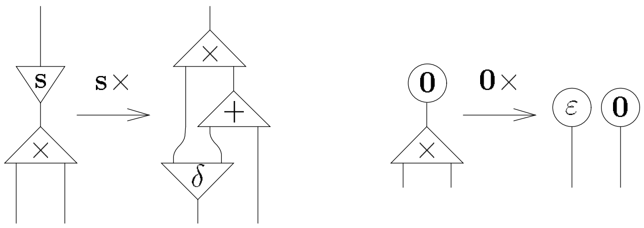


One gets the following rules for addition:

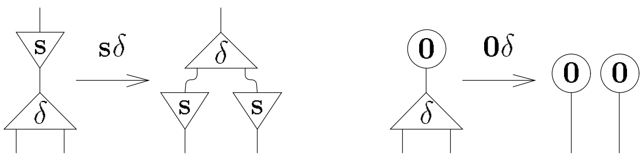


Note that the argument y is used twice in the first equation defining the multiplication, and it is not used at all in the second one. For that reason, two extra symbols are needed: δ (*duplicator*) of arity 2 and ε (*eraser*) of arity 0, with the following interpretation:

The idea is that a net representing a natural number should be duplicated when it is connected to the principal port of a δ, and it should be erased when it is connected to the principal port of an ε. One gets the following rules for multiplication:

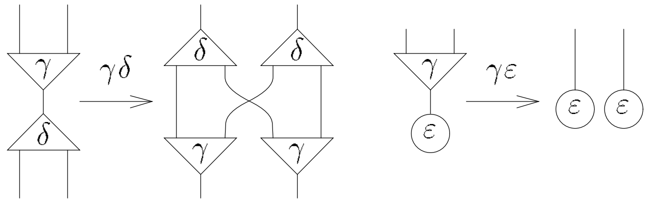


Of course, extra rules are needed for duplication and erasing:



An example of computation is given in figure 1. Note that this computation is essentially parallel, since in many cases, several rules can be applied simultaneously.

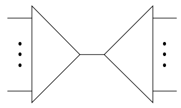
If one had to duplicate tree-like structures, such as lists of natural numbers, then one would introduce the following rules for each binary constructor γ:



We shall find them again in the system of interaction combinators.

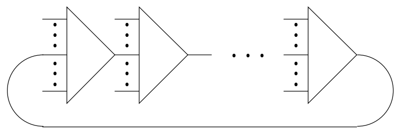
1.6 Reduced nets

Two kinds of configuration may lock a computation: the *irreducible cuts* and the *vicious circles*. A *cut* consists of two cells connected through their principal ports:



For a given interaction system, there are two kinds of cuts: the *reducible* ones and the *irreducible* ones. For instance, in the case of unary arithmetics, a cut between **s** and $+$ is reducible, whereas a cut between **s** and **0** is irreducible. If an irreducible cut occurs in a net ν, it can never be eliminated, because the cells can only interact through their principal ports.

A *principal path* of length n consists of n cells $c_1, \ldots, c_n$ such that the principal port of c_i is connected to an auxiliary port of c_{i+1}. In particular, a wire can be considered as a principal path of length 0. A *vicious circle* of length n is a closed principal path of length n:



In particular, a cyclic wire can be considered as a vicious circle of length 0. Clearly, a vicious circle can never be eliminated, because no c_i can interact first.

Say that a net is *reduced* if it contains no cut and no vicious circle. Trees and wirings are typical examples of

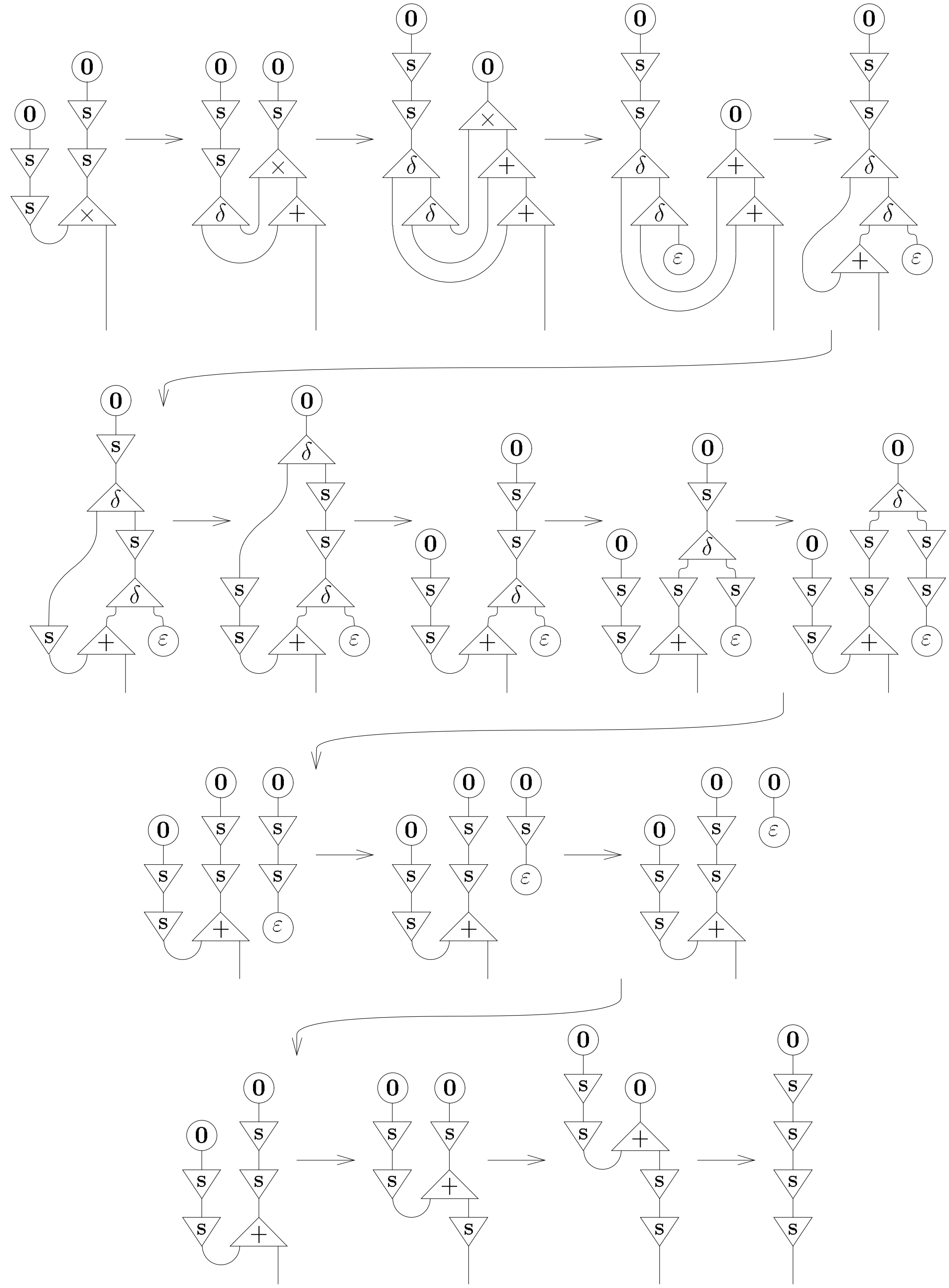
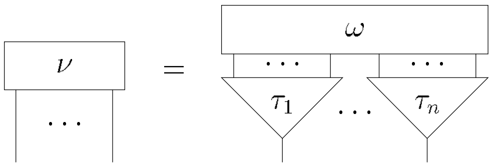


Figure 1: $2 \times 2 = 4$

reduced nets. An irreducible net is not necessarily reduced, since it may contain irreducible cuts or vicious circles, but a reduced net is always irreducible.

Note that in the case of unary arithmetics, for instance, all right members of rules are reduced nets. From now on, we shall only consider interaction systems satisfying this extra condition. If it is not the case, one can always try to reduce the right members. If this does not work, it means that there is something wrong in the system.

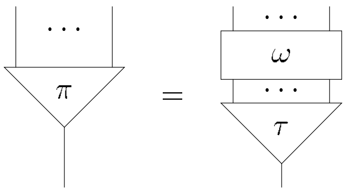
Proposition 2 *Any reduced net ν with n free ports can be uniquely decomposed as follows:*



where $\tau_1, \ldots, \tau_n$ are trees and ω is a wiring.

Proof: By induction on the number of cells. If there is no cell, then ν is a wiring. Otherwise, one can choose a cell c_1, and construct a principal path $c_1, c_2, c_3, \ldots$ Since there is no cut and no vicious circle, this path must stop at some cell c_n whose principal port is connected to a free port. It suffices to apply the induction hypothesis to the net obtained by removing this c_n, which is also a reduced net. **Q.e.d.**

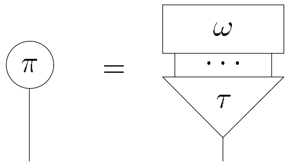
One says that a reduced net is *principal* if it is a single wire (degenerate case), or if one of its free ports is connected to a principal port and all other free ports are connected to auxiliary ports. For instance, a tree is a principal net. Proposition 2 gives the following decomposition for a principal net π:



where τ is a tree and ω is a wiring which does not connect upper ports. If π has $n+1$ ports, one says that π is a principal net *of arity n*, because it can be considered as a generalized symbol of arity n. In the non-degenerate case, the free port which is connected to a principal port is called the *root* of π. In the degenerate case, the choice of the root is arbitrary. We shall need a result which follows easily from the above decomposition:

Lemma 1 *Let π be a principal net with $n+1$ free ports x_0, x_1, …, x_n, the root being x_0. Then, for $i = 1, \ldots, n$, there is a unique principal path from x_i to x_0. Except in the degenerate case, there is no other principal path between free ports.*

By proposition 2, a reduced net with no free port is necessarily empty. A reduced net with only one free port is called a *package*. Such a net is necessarily principal, with the following decomposition:



where τ is a tree and ω is a wiring.

1.7 Translations

Let Σ and Σ' be alphabets of symbols with arities. A *translation* Φ of Σ into Σ' maps each symbol α of arity n in Σ to a principal net $\Phi(\alpha)$ of arity n, built with symbols of Σ'. If furthermore, this $\Phi(\alpha)$ is non-degenerate for every α, one says that Φ is *strict*. A translation extends to nets in an obvious way. Just notice that, in the non-strict case, the translation may create cyclic wires, and this is another reason why cyclic wires have been allowed in the definition of nets. The fact that the $\Phi(\alpha)$ are principal has the following consequence:

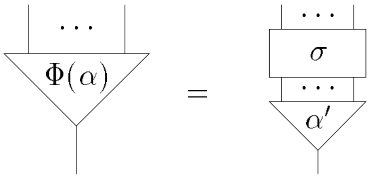
Proposition 3 *If ν is a reduced net, then $\Phi(\nu)$ is a reduced net. The converse holds if Φ is strict.*

Proof: It suffices to check the following points:

- if $\Phi(\nu)$ contains a cut, so does ν;
- if Φ is strict and ν contains a cut, so does $\Phi(\nu)$;
- if $\Phi(\nu)$ contains a vicious circle, then ν contains a vicious circle or a cut;
- if ν contains a vicious circle, so does $\Phi(\nu)$.

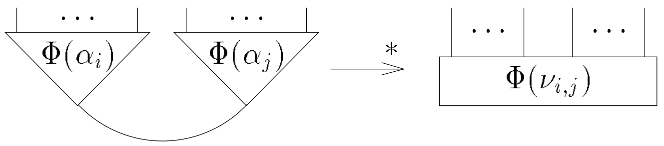
Lemma 1 is used for the last two statements. One simply has to be careful in the non-strict case. For instance, it may happen that $\Phi(\nu)$ contains a vicious circle, and ν contains no vicious circle, but a cut. It may also happen that ν contains a cut, and $\Phi(\nu)$ is reduced. **Q.e.d.**

Composition of translations is defined in the obvious way. One says that Φ is *invertible* if there is a translation Φ^{-1} of Σ' into Σ such that $\Phi^{-1} \circ \Phi = \mathrm{id}_\Sigma$ and $\Phi \circ \Phi^{-1} = \mathrm{id}_{\Sigma'}$. In that case, it is easy to see that each $\Phi(\alpha)$ is necessarily of the following form:



where α' is a symbol of Σ' and σ is a permutation. In other words, Φ is given by a renaming of the symbols and a permutation of the auxiliary ports of each symbol.

Assume that some interaction system is given for Σ and similarly for Σ'. One says that Φ is a *translation of interaction system* if it is compatible with reduction. This is expressed by the following property:



The $\to^*$ stands for a reduction of arbitrary length, possibly 0. Remember that $\nu_{i,j}$ is supposed to be reduced, and so is $\Phi(\nu_{i,j})$ by proposition 3. Therefore, if Φ is strict, the left member is not reduced, and the reduction cannot be of length 0. Similarly, if Φ is invertible, the reduction must be of length 1. In general, let L (resp. M) be the minimal (resp. the maximal) length of all those reductions. By the above remark, $L > 0$ if Φ is strict, and $L = M = 1$ if Φ is invertible. Obviously, one has:

Proposition 4 *If μ is a net built with symbols of Σ which reduces in n steps to ν, then $\Phi(\mu)$ reduces in n' steps to $\Phi(\nu)$, with $Ln \leq n' \leq Mn$.*

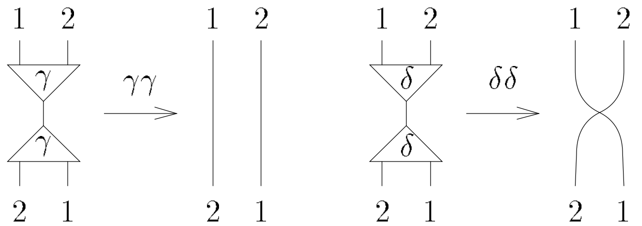
2 Interaction combinators

This section is the heart of the paper. It is entirely devoted to the universality of the system of interaction combinators.

2.1 The system

The *system of interaction combinators* consists of 3 symbols, called *combinators*: γ (*constructor*) and δ (*duplicator*) of arity 2, and ε (*eraser*) of arity 0. The 6 interaction rules (figure 2) are of two kinds: *commutation* when the two cells carry different symbols ($\gamma\delta$, $\gamma\varepsilon$, $\delta\varepsilon$) and *annihilation* when they carry the same symbol ($\gamma\gamma$, $\delta\delta$, $\varepsilon\varepsilon$). The reader can check that the symmetry condition is satisfied.

Note that the annihilations for γ and δ are not the same. Furthermore, if one numbers the auxiliary ports, one realizes that it is $\gamma\gamma$, not $\delta\delta$, which exchanges the ports:



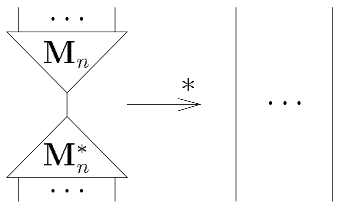
One can already notice that the process of reduction does not always terminate. Figure 3 shows a net which reduces to itself in 4 steps. In fact, we shall prove that, despite its simplicity, this system is universal in the following sense:

Theorem 1 *Any interaction system can be translated into the system of interaction combinators.*

In order to prove this, we shall introduce several constructions which are inspired by the rules of linear logic: the *multiplexors* and the *transpositors* (inspired by the multiplicative rules), the *menus* and the *selectors* (inspired by the additive rules), the *codes*, the *copiers*, and the *decoder* (inspired by the exponential rules).

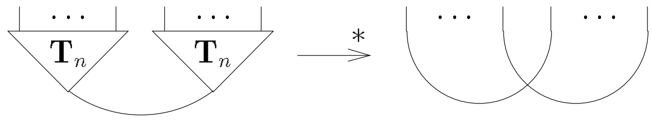
2.2 Multiplexors and transpositors

For any $n \in \mathbf{N}$, one constructs two principal nets $\mathbf{M}_n$ and $\mathbf{M}_n^*$ (*multiplexors*) of arity n, with the following property:



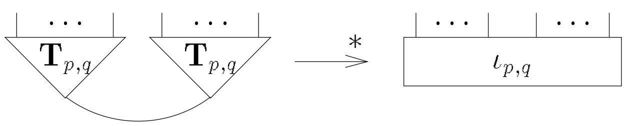
An implementation of these multiplexors is given in figure 4. The needed rules are $\gamma\gamma$ and $\varepsilon\varepsilon$. There is an alternative implementation using δ instead of γ, but in fact, it will be essential that our implementation of the multiplexors does not use δ.

We shall also need a kind of autodual multiplexor, that is a principal net $\mathbf{T}_n$ of arity n with the following property:



An implementation of these autodual multiplexors is given in figure 5. The needed rules are $\delta\delta$ and $\varepsilon\varepsilon$. Note that γ is not suitable for that purpose.

More generally, if $p, q \in \mathbf{N}$, one constructs a principal net $\mathbf{T}_{p,q}$ (*transpositor*) of arity $2p + q$ with the following property:



where $\iota_{p,q}$ is the involutive permutation of $\{1, \ldots, 2p + q\}$ which exchanges 1 with 2, 3 with 4, and so on until $2p$.

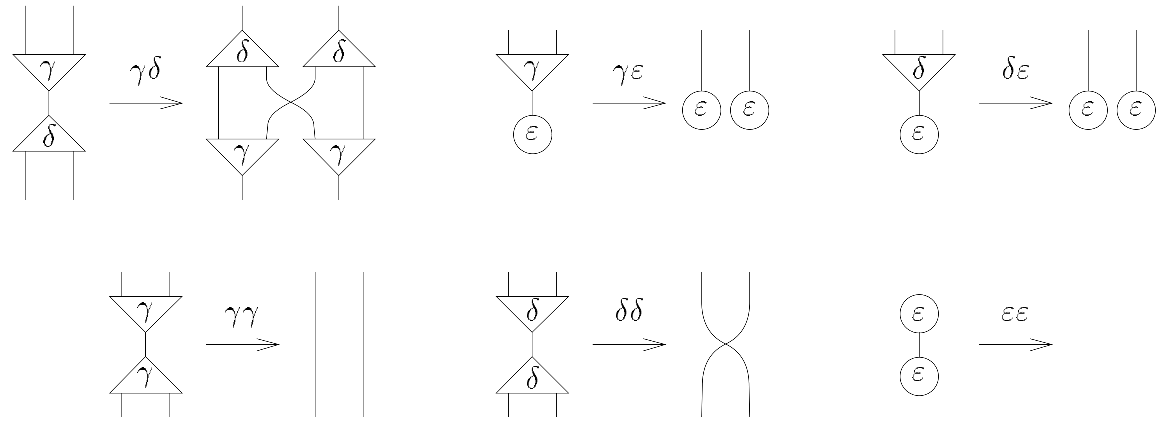


Figure 2: interaction rules for the combinators

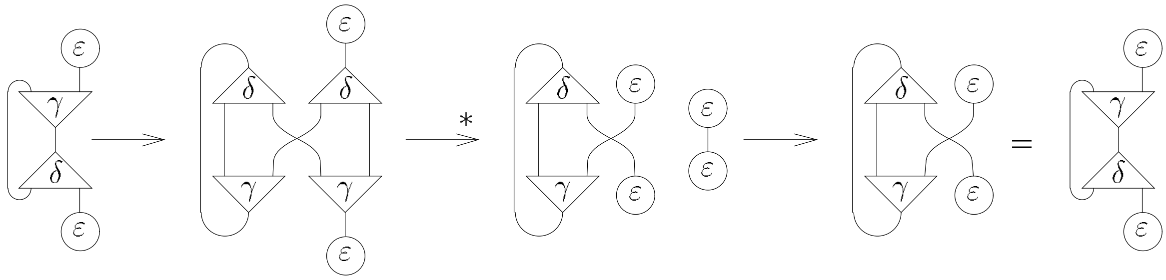


Figure 3: a nonterminating computation

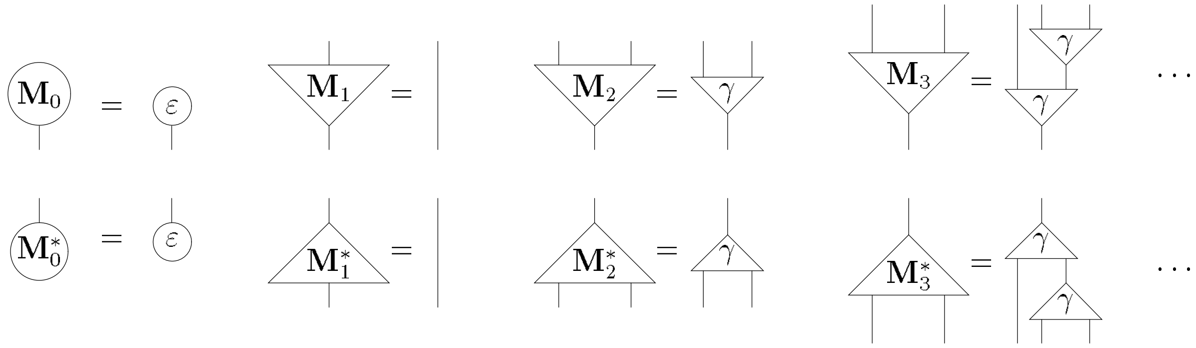


Figure 4: implementation of the multiplexors

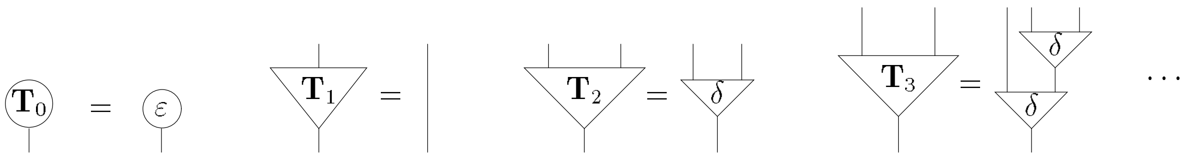
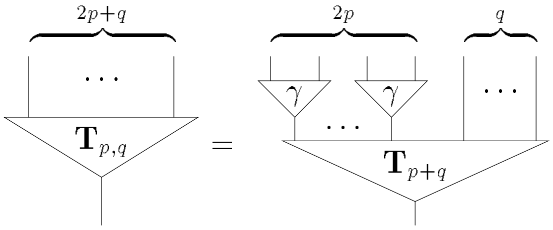


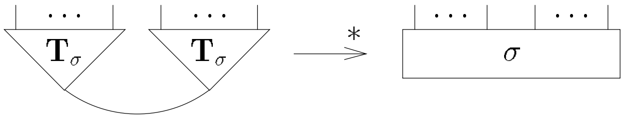
Figure 5: implementation of the autodual multiplexors

Here is a possible implementation of these transpositors:

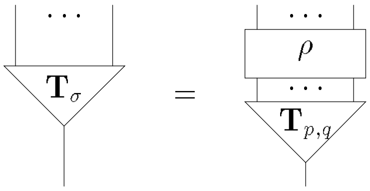


The needed rule is $\gamma\gamma$. Note that δ is not suitable for that purpose.

Finally, if σ is an involutive permutation of $\{1, \ldots, n\}$, one constructs a principal net $\mathbf{T}_\sigma$ of arity n with the following property:



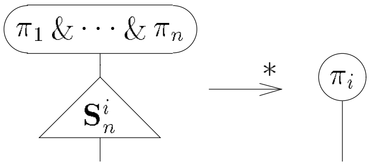
Any such permutation is indeed a product of disjoint transpositions, that is $\sigma = \rho^{-1} \circ \iota_{p,q} \circ \rho$ where ρ is a permutation of $\{1, \ldots, n\}$ and $2p + q = n$. So, here is a possible implementation of $\mathbf{T}_\sigma$:



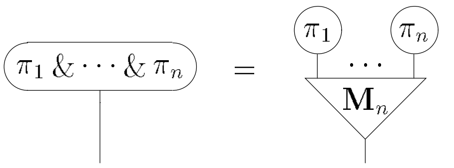
Note that conversely, if such a $\mathbf{T}_\sigma$ exists, the permutation σ is necessarily involutive. This is a consequence of the symmetry condition.

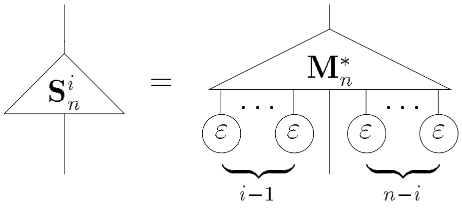
2.3 Menus and selectors

If $\pi_1, \ldots, \pi_n$ are packages, one constructs a new package $\pi_1 \,\&\cdots\&\, \pi_n$ (*menu*) from which the original ones can be extracted by means of principal nets $\mathbf{S}^1_n, \ldots, \mathbf{S}^n_n$ (*selectors*) of arity 1:

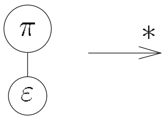


Here is a possible implementation:





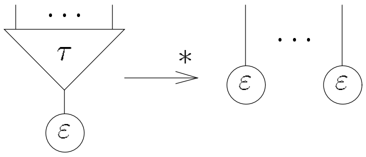
One uses the fact any package π can be erased as follows:



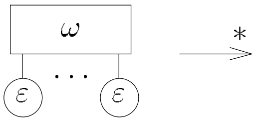
This is an obvious corollary of the following lemma:

Lemma 2 *(erasing)*

i) For any tree τ:



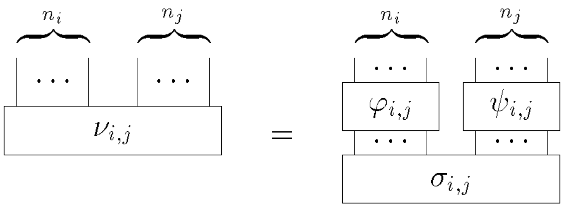
ii) For any wiring ω:



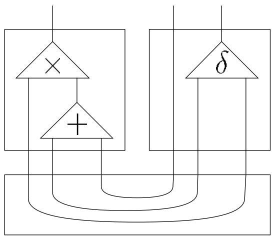
Both statements are proved by induction, using $\gamma\varepsilon$, $\delta\varepsilon$ and $\varepsilon\varepsilon$. In fact, (i) holds more generally for principal nets and (ii) for reduced nets.

2.4 Translation (restricted case)

With the previous constructions, we can already simulate interaction systems which are in some sense *recursion-free*. Consider an interaction system with m symbols $\alpha_1, \ldots, \alpha_m$ of respective arities $n_1, \ldots, n_m$. The right member of a rule is a reduced net which can be decomposed as follows:



where $\varphi_{i,j}$ and $\psi_{i,j}$ are reduced nets, and $\sigma_{i,j}$ is a permutation. Of course, such a decomposition is not unique, but there is a canonical one. For instance, here is the canonical decomposition for the rule $\mathbf{s}\times$ of section 1.5:



By the symmetry condition, and by unicity of the canonical decomposition, one gets $\varphi_{j,i} = \psi_{i,j}$ and $\sigma_{j,i} = \sigma^{-1}_{i,j}$. In particular, if $i = j$, one gets $\sigma_{i,i} = \sigma^{-1}_{i,i}$, which means that $\sigma_{i,i}$ is involutive. If $i \neq j$, it is always possible to include the permutation into $\varphi_{i,j}$ (or into $\varphi_{j,i}$), so that the decomposition is no longer canonical, but $\sigma_{i,j}$ is an identity. To sum up, one can assume that in all cases, $\varphi_{j,i} = \psi_{i,j}$, $\sigma_{j,i} = \sigma_{i,j}$ and $\sigma_{i,j}$ is involutive.

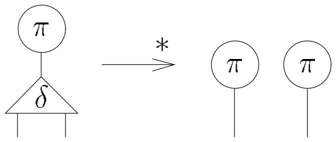
Now, assume that the alphabet $\{\alpha_1, \ldots, \alpha_m\}$ can be ordered in such a way that the net $\varphi_{i,j}$, when defined, contains only symbols which are strictly smaller than α_j (and symmetrically, $\psi_{i,j} = \varphi_{j,i}$ contains only symbols which are strictly smaller than α_i). It is easy to see that the process of reduction always terminates in such a system. A typical example is our system of interaction combinators without the rule $\gamma\delta$ (using an ordering such that $\varepsilon < \gamma$ and $\varepsilon < \delta$, and a non-canonical decomposition for $\gamma\varepsilon$ and $\delta\varepsilon$).

With this restriction, a principal net $[\alpha_i]$ can be defined inductively for each symbol α_i as in figure 6. In this definition, $[\varphi_{i,j}]$ stands for the net obtained by replacing each α_k by $[\alpha_k]$ in $\varphi_{i,j}$. If $\varphi_{i,j}$ happens to be undefined, which means that there is no rule for α_i, α_j, then $\Gamma_{i,j}$ can be fixed arbitrarily. In that way, one gets a translation of interaction systems (figure 7).

2.5 Duplication

Intuitively, the package Γ_i above represents the finite tree of all possible futures of a cell. In the general case, this tree may be infinite, but it is possible to replace this actual infinity by a potential one: it suffices to use something like a genetic code. Duplication is clearly essential for this purpose.

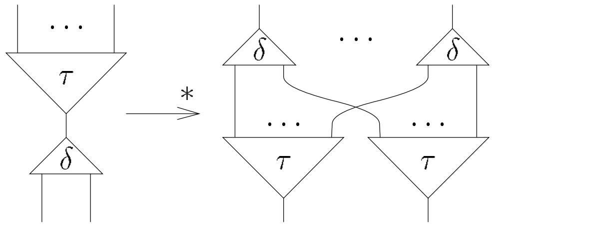
Unfortunately, the analogue of lemma 2 for duplication does not hold, because δ cannot duplicate δ (whereas ε can erase ε). Nevertheless, a package π without δ can be duplicated as follows:



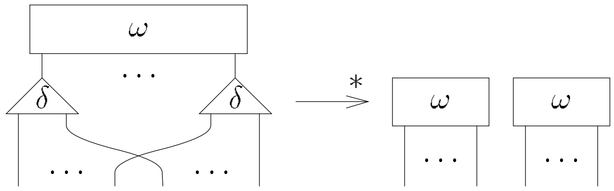
This is an obvious corollary of the following lemma:

Lemma 3 *(duplication)*

i) For any tree τ without δ:



ii) For any wiring ω:

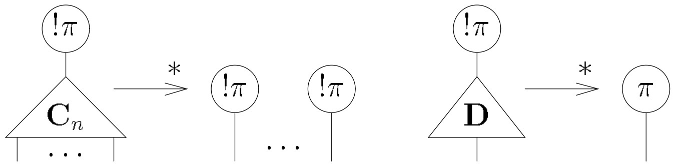


Both statements are proved by induction, using $\gamma\delta$, $\delta\varepsilon$ and $\delta\delta$. Note that the rule $\delta\delta$ is forced by (ii), and this is why δ cannot duplicate δ. Again, (i) holds more generally for principal nets and (ii) for reduced nets without δ.

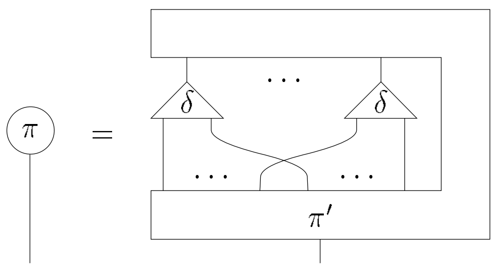
How to cope with this impossibility of duplicating arbitrary packages? This is the crucial point of our proof.

2.6 Codes, copiers and decoder

One constructs, for any package π, another package $!\pi$ (*code* of π) which can be duplicated, and from which π can be extracted. More precisely, one constructs principal nets $\mathbf{C}_n$ (*copier*) of arity n and $\mathbf{D}$ (*decoder*) of arity 1, with the following properties:



If one manages to construct $!\pi$ without using δ, then by lemma 3, there is an obvious implementation of the $\mathbf{C}_n$ (in fact, the same as for the $\mathbf{T}_n$). It remains to define $!\pi$ and $\mathbf{D}$. If π contains n occurrences of δ, it can be decomposed as follows:



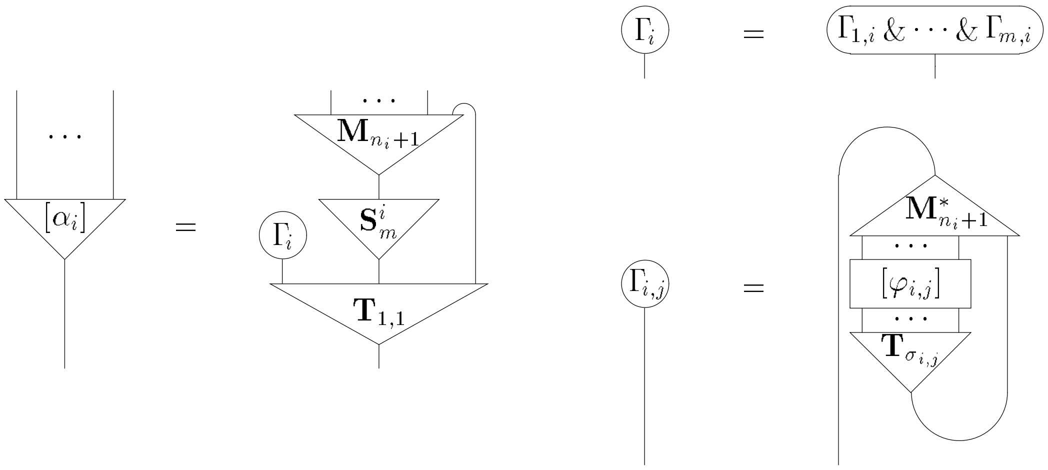


Figure 6: translation of a cell (restricted case)

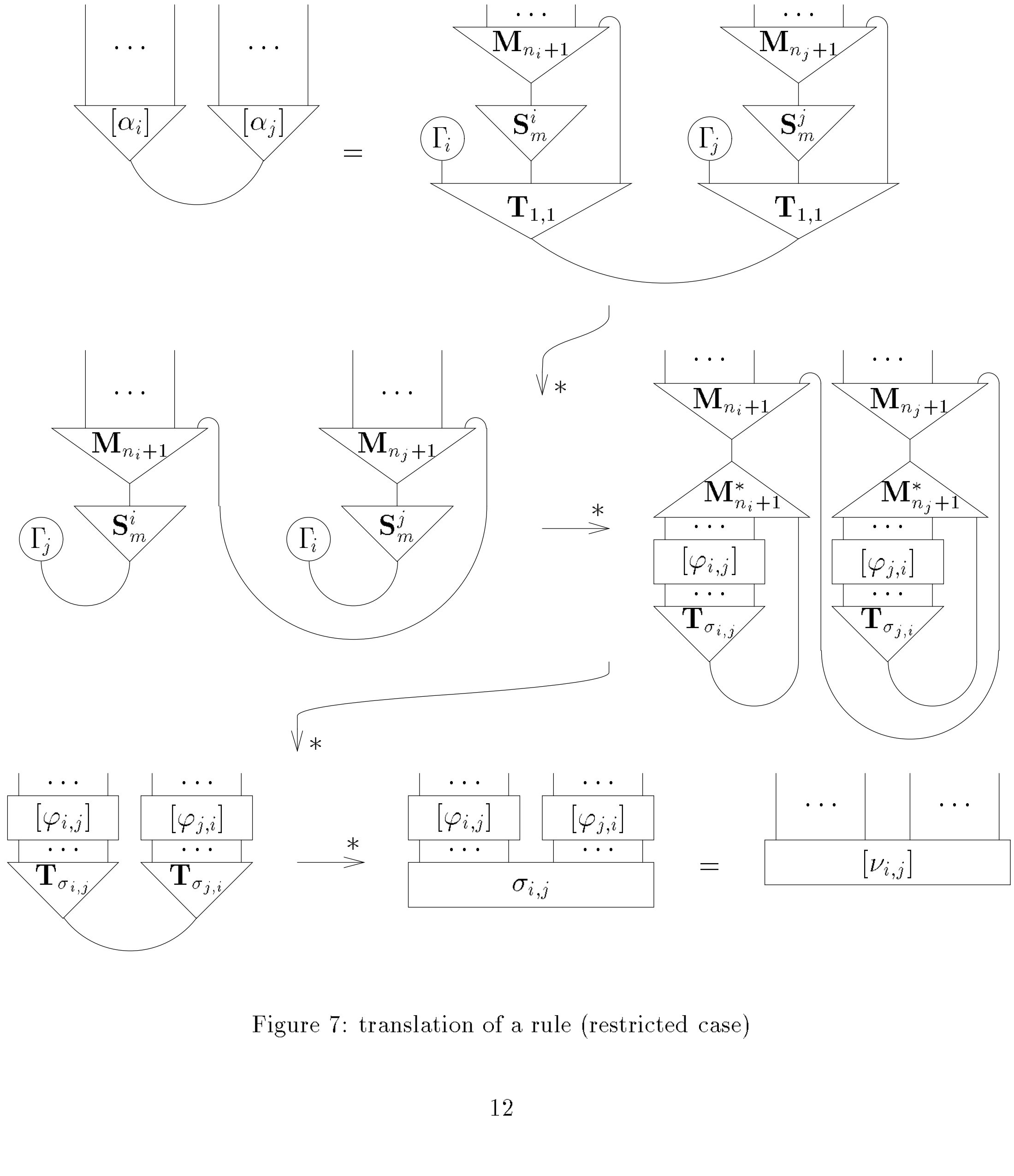
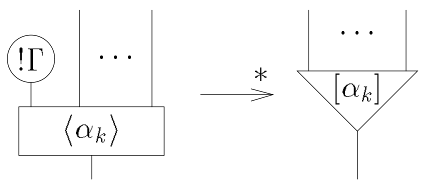


Figure 7: translation of a rule (restricted case)

where π' is a reduced net with $3n+1$ free ports containing no δ. An implementation of the code and the decoder is given in figure 8. Clearly, $!\pi$ contains no δ and π can be extracted from it by means of the decoder (figure 9). Note that lemma 3 is used once more.

2.7 Translation (general case)

Using the previous constructions, a principal net $[\alpha_i]$ can be defined for each symbol α_i as in figure 10, where $p_{i,j}$ is the total number of cells in $\varphi_{i,j}$ and $\langle\varphi_{i,j}\rangle$ is obtained by replacing each symbol α_k in $\varphi_{i,j}$ by $\langle\alpha_k\rangle$. Note that this $\langle\alpha_k\rangle$ has one more port than the cell α_k. All those extra ports must be plugged into the copier $\mathbf{C}_{p_{i,j}}$ in $\Gamma_{i,j}$. Otherwise, this translation works very much like the previous one. The package $!\Gamma$ plays the role of a genetic code, from which each $[\alpha_k]$ can be extracted as follows:



The reader is invited to complete the proof of theorem 1 by working out the analogue of figure 7 for this translation.

2.8 Minimality of the system

One may wonder whether the system of interaction combinators is minimal. In other words, is it possible to remove symbols or rules without losing the property of theorem 1? One can already make the following remarks:

- $\gamma\delta$ is necessary because it is the only rule which increases the number of cells. In particular, γ and δ are necessary;
- $\varepsilon\varepsilon$ is necessary because it is the only rule for which the right member is empty. In particular, ε is necessary;
- one rule among $\gamma\varepsilon$ and $\delta\varepsilon$ is necessary because they are the only rules which increase the number of occurrences of ε;
- one rule among $\gamma\gamma$ and $\delta\delta$ is necessary because they are the only rules for which the right member is a nonempty wiring.

A more detailed study shows that, in fact, both $\gamma\gamma$ and $\delta\delta$ are necessary. On the other hand, it happens that the system remains universal if one removes $\delta\varepsilon$, but this is not a significant simplification.

One may also wonder whether there is a universal interaction system with less symbols. Starting from the interaction combinators, Denis Bechet proposed a universal interaction system with only 2 symbols and 3 rules (private communication), but one of his rules is quite complicated. Also, it is easy to see that one symbol is not enough. Therefore, we conjecture that our system of interaction combinators is essentially the simplest one satisfying theorem 1.

3 Semantics of combinators

Here, we interpret the nets of combinators as reversible 2-stack machines. This interpretation gives a notion of equivalence on nets of combinators, and suggests a system of directed combinators.

3.1 Execution

Let ν be a net of combinators which reduces to some wiring ω, as in figure 11. Looking carefully at the rules of interaction, one sees that it is possible to compute ω without reducing ν, just by travelling in ν. This will be called the *execution* of ν.

During this execution, one carries a pair (u, v) where u (the *γ-stack*) and v (the *δ-stack*) are finite strings over the alphabet $\{\mathbf{p}, \mathbf{q}\}$. We shall write 1 for the empty string. The letter $\mathbf{p}$ (resp. $\mathbf{q}$) corresponds to the first (resp. the second) auxiliary port of γ or δ. One travels according to the following rules (figure 12):

- if one enters a cell through an auxiliary port, one pushes the corresponding letter onto the relevant stack, and one exits through the principal port;
- if one enters a cell through the principal port, one pops $\mathbf{p}$ or $\mathbf{q}$ from the relevant stack, and one exits through the corresponding auxiliary port (in the case of a δ-cell) or through the other one (in the case of a γ-cell). If the relevant stack is empty, one stops.

Of course, one considers that the *ε-stack* is always empty. An example of execution is given in figure 13.

Let ν be an arbitrary net of combinators. One writes $(x, u, v) \overset{\nu}{\rightsquigarrow} (x', u', v')$ if, starting from the free port x with the pair (u, v), one eventually reaches the free port x' with the pair (u', v'). One writes $\overline{u}$ for the string obtained by exchanging $\mathbf{p}$ with $\mathbf{q}$ in u.

Proposition 5 *The execution satisfies the following properties:*

i) for any (x, u, v), there is at most one (x', u', v') such that $(x, u, v) \overset{\nu}{\rightsquigarrow} (x', u', v')$ (determinism);

ii) if $(x, u, v) \overset{\nu}{\rightsquigarrow} (x', u', v')$ then $(x', \overline{u'}, v') \overset{\nu}{\rightsquigarrow} (x, \overline{u}, v)$, and conversely (reversibility);

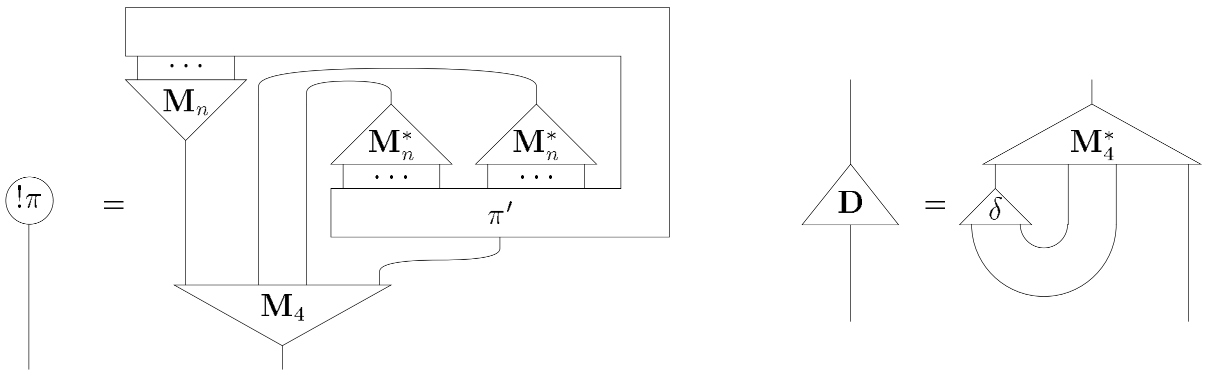


Figure 8: implementation of the code and the decoders

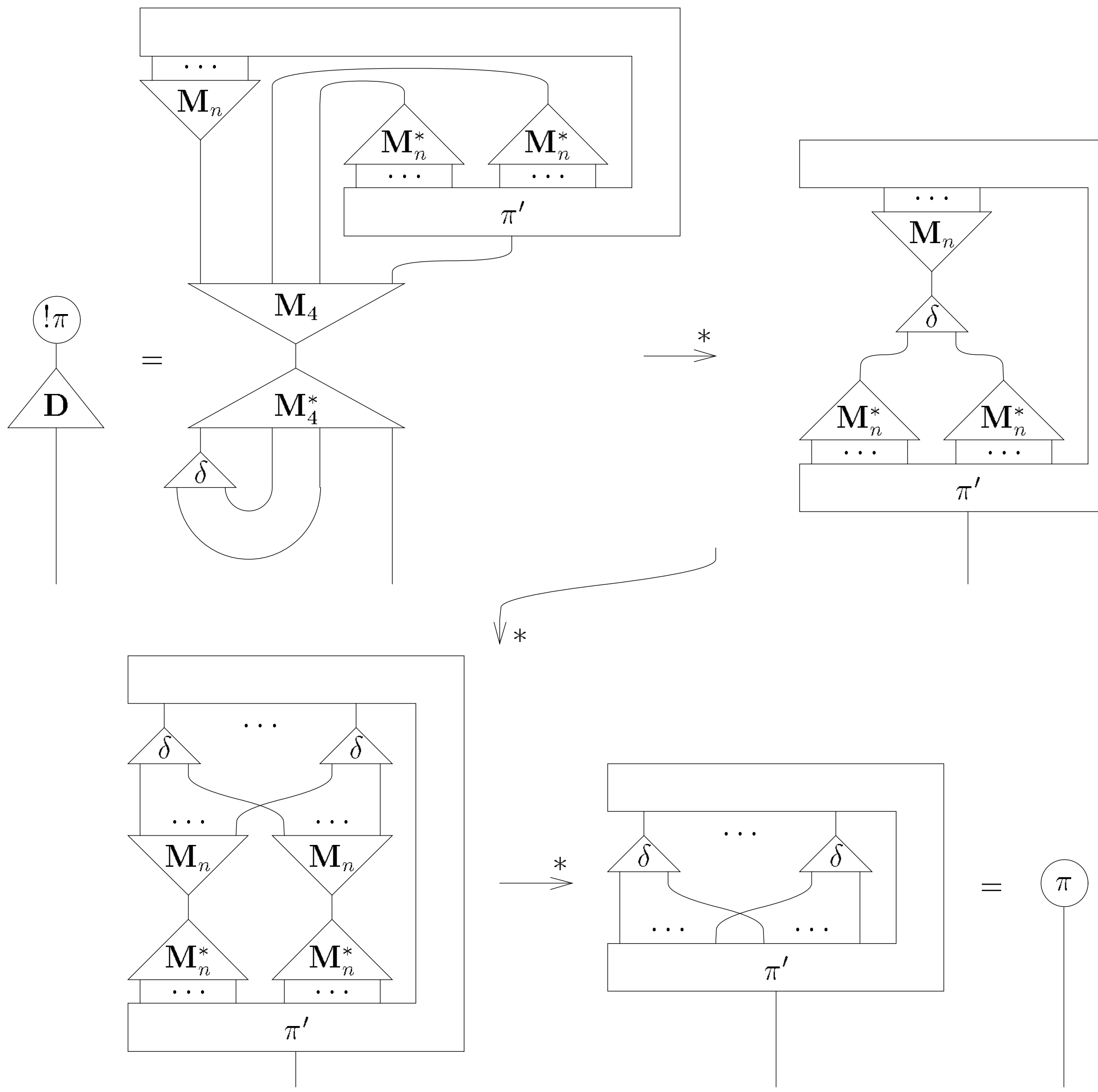


Figure 9: decoding

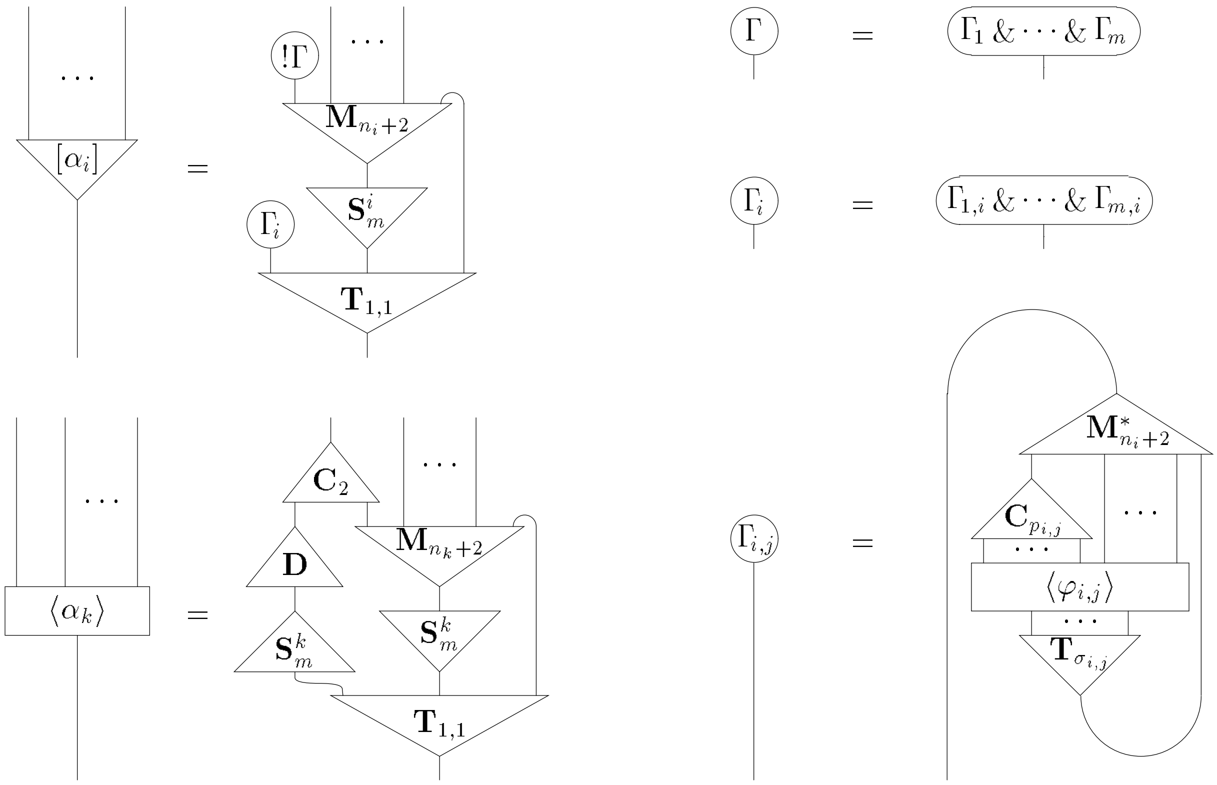


Figure 10: translation of a cell (general case)

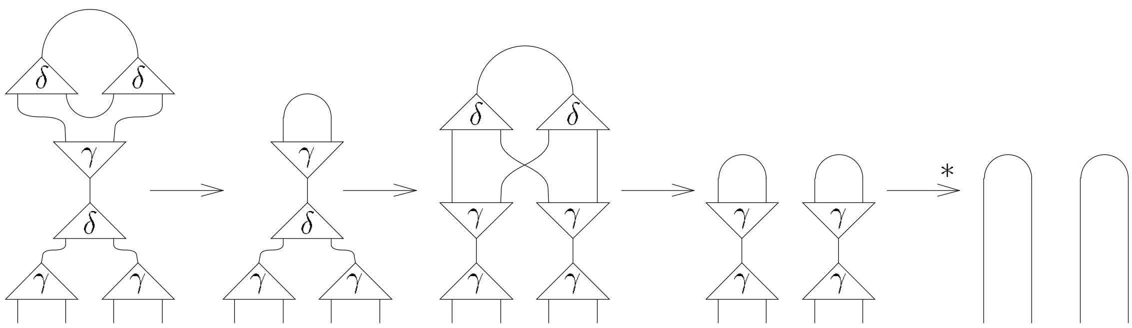


Figure 11: a net reducing to a wiring

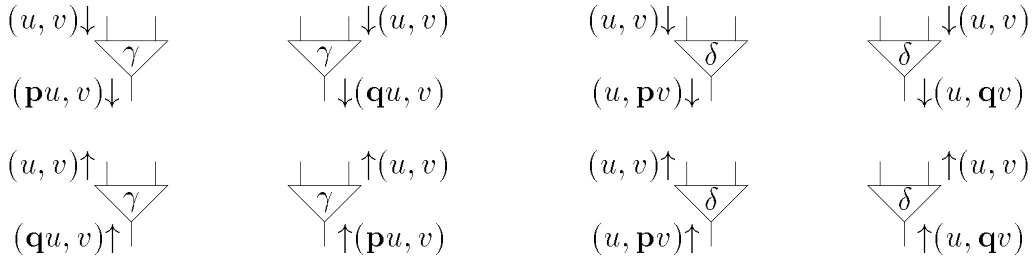


Figure 12: rules of execution

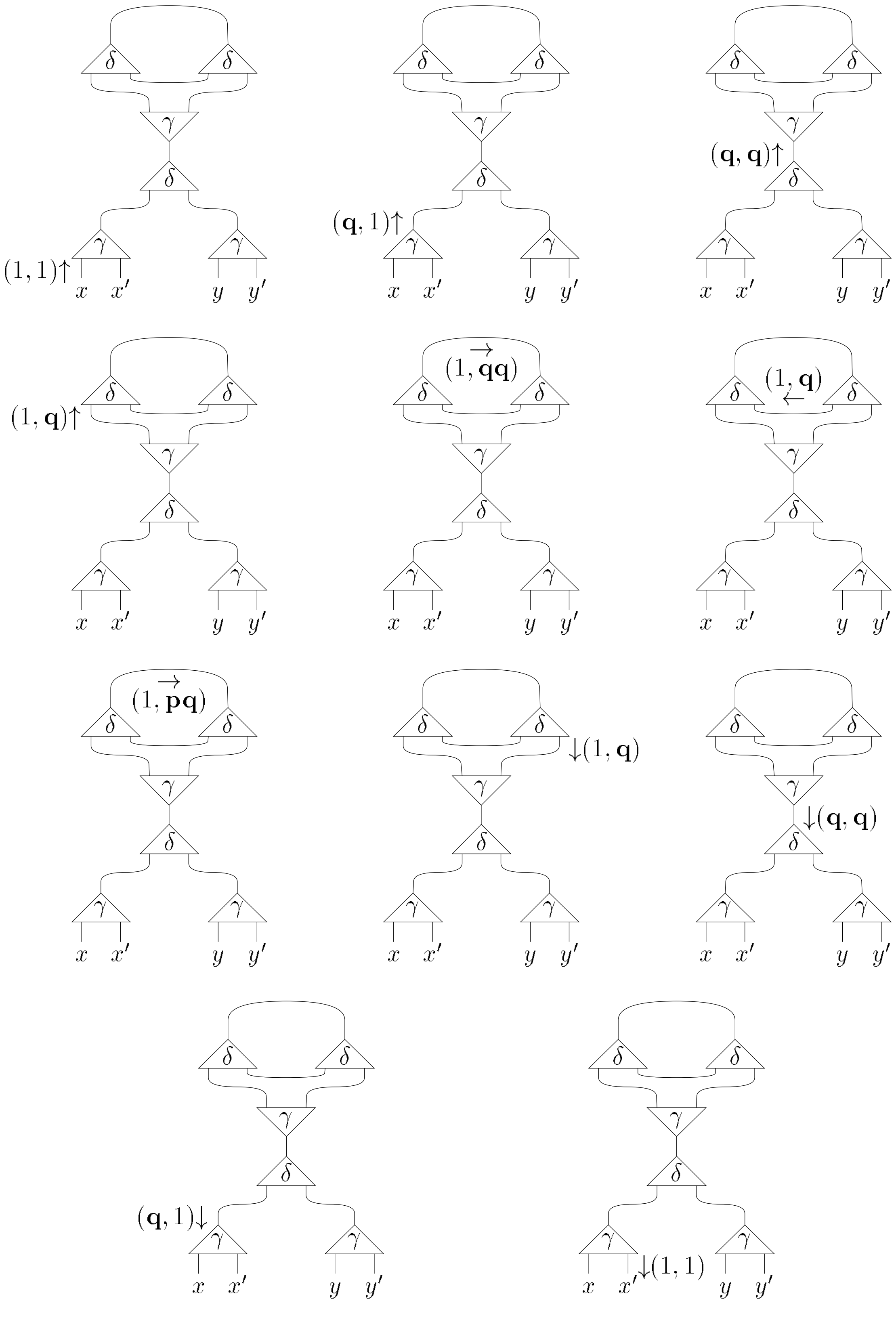
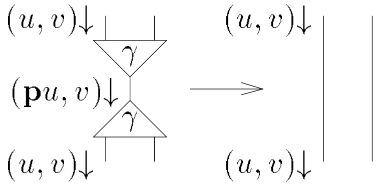


Figure 13: example of execution

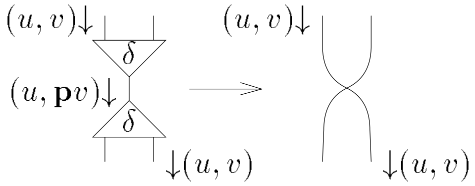
iii) if $(x,u,v) \overset{\nu}{\rightsquigarrow} (x',u',v')$, then for any u'' and v'', $(x,uu'',vv'') \overset{\nu}{\rightsquigarrow} (x',u'u'',v'v'')$ (monotonicity);

iv) if μ reduces to ν, then $\overset{\mu}{\rightsquigarrow}$ and $\overset{\nu}{\rightsquigarrow}$ are the same (invariance).

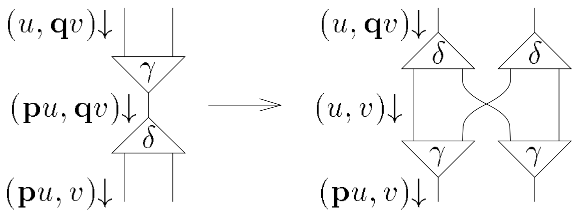
Proof: The first three statements are obvious. The fact that u becomes $\overline{u}$ in the reverse computation comes from our definition of the execution in the case of a γ-cell. It is of course related to the fact that $\gamma\gamma$ exchanges the ports. For the last statement, it suffices to check that the execution is invariant by each rule. Clearly, the execution has been defined in such a way that it is invariant by $\gamma\gamma$ and $\delta\delta$. Here is a typical case for $\gamma\gamma$:



Here is a typical case for $\delta\delta$:



The execution is also invariant by $\gamma\delta$, because the two stacks are independent. Here is a typical case:



The remaining cases are similar. **Q.e.d.**

3.2 Equivalence

Say that two nets of combinators μ and ν with the same free ports are *equivalent* if $\overset{\mu}{\rightsquigarrow}$ and $\overset{\nu}{\rightsquigarrow}$ are the same. This is obviously a congruence: if one replaces a subnet of ν by an equivalent one, the resulting net is equivalent to ν. Furthermore, the invariance property tells us that μ and ν are equivalent whenever μ reduces to ν.

In the case of a wiring ω, one has $(x,1,1) \overset{\omega}{\rightsquigarrow} (x',1,1)$ if and only if x is connected to x' in ω. Therefore, if a net ν reduces to a wiring ω as in figure 11, this ω can be obtained by executing ν, as in figure 13. Note also that two wirings are equivalent if and only if they are equal. In order to characterize the equivalence of reduced nets, we shall use the rules of figure 14, which can be applied in both directions.

Say that a free port x of a reduced net ν *accepts* a pair (u,v) if there is a free port x' and a pair (u',v') such that $(x,u,v) \overset{\nu}{\rightsquigarrow} (x',u',v')$. Of course, this notion depends only on the equivalence class of ν. Say that x is *passive* if it is not connected to a principal port, and say that x is γ-*active* (resp. δ-*active*, ε-*active*) if ν can be transformed by means of the rules of figure 14, in such a way that x is connected to the principal port of a γ-cell (resp. a δ-cell, an ε-cell). Again, these notions depend only on the equivalence class of ν:

- x is passive if and only if it accepts all (u,v);
- x is γ-active if and only if it accepts no $(1,v)$;
- x is δ-active if and only if it accepts no $(u,1)$;
- x is ε-active if and only if it accepts no (u,v).

The first statement follows easily from the decomposition of proposition 2, and the other ones are proved by induction on the number of cells.

Proposition 6 *If μ and ν are reduced nets of combinators with the same free ports, then μ is equivalent to ν if and only if μ can be transformed into ν by means of the rules of figure 14.*

Proof: It is easy to see that the execution is invariant by these rules. It remains to show that, if μ is equivalent to ν, then μ can be transformed into ν by means of the rules of figure 14. This is proved by induction on the number of cells in ν. If all free ports are passive, then μ and ν are the same wiring. Otherwise, there is a free port x of ν which is connected to the principal port of a cell, for instance a γ-cell. In other words, x is γ-active, and μ can be transformed in such a way that x is connected to the principal port of a γ-cell. One can remove the γ-cell in both cases, and apply the induction hypothesis to the remaining nets. **Q.e.d.**

3.3 Algebraic formulation

The execution can be reformulated in a more algebraic way, as in the *geometry of interaction* (see [Gir89, DR95]). For conciseness, we omit all proofs here.

First, one considers the non-commutative ring $\Re$ generated by the letters $\mathbf{p}$, $\mathbf{q}$, $\mathbf{p}^*$ and $\mathbf{q}^*$ subjected to the following equations:

$$\mathbf{p}^*\mathbf{p} = \mathbf{q}^*\mathbf{q} = 1, \qquad \mathbf{p}^*\mathbf{q} = \mathbf{q}^*\mathbf{p} = 0.$$

The letters $\mathbf{p}$ and $\mathbf{q}$ are called *positive*, and the other ones are called *negative*. A *reduced monomial* is an expression of the form uu' where u consists of positive letters and u' consists of negative ones. By the above equations, the product of two such monomials is either a reduced monomial or 0. For instance:

$$(\mathbf{pp}^*\mathbf{q}^*)(\mathbf{qppq}^*) = (\mathbf{pp}^*)(\mathbf{ppq}^*) = \mathbf{ppq}^*,$$

$$(\mathbf{pp}^*\mathbf{q}^*)(\mathbf{qqpq}^*) = (\mathbf{pp}^*)(\mathbf{qpq}^*) = 0.$$

So, one can show that any element of $\Re$ can be uniquely written as a finite sum of reduced monomials. In particular, the strings of positive letters can be identified with the corresponding monomials in $\Re$. Moreover, the duality can be extended to the whole ring $\Re$, in such a way that it satisfies the following properties:

$$(uv)^* = v^*u^*, \qquad 1^* = 1, \qquad (u+v)^* = u^* + v^*, \qquad 0^* = 0,$$

$$u^{**} = u, \qquad u^*u = 1.$$

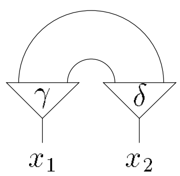
Let ν be an arbitrary net of combinators. If x and x' are free ports of ν, a *direct path* P from x to x' is a path from x to x' such that, if one enters through an auxiliary port (resp. the principal port), then one exits through the principal port (resp. an auxiliary port). For each type of crossing, one introduces an element of the ring $\Re \otimes \Re$ as in figure 15. The *weight* $|P|$ is the product of all these elements, in reverse order. For instance, the weight of the path in figure 13 is

$$(\mathbf{q}^*\mathbf{q}\mathbf{q}^*\mathbf{q}) \otimes (\mathbf{q}^*\mathbf{p}^*\mathbf{p}\mathbf{q}^*\mathbf{q}\mathbf{q}) = 1 \otimes 1$$

The weight is invariant by reduction. In fact, if one considers the interaction rules for γ and δ, it appears that the equations defining $\Re$ correspond to the rules of annihilation ($\gamma\gamma$ and $\delta\delta$) and the commutation $(u \otimes 1)(1 \otimes v) = (1 \otimes v)(u \otimes 1)$ to the rule of commutation ($\gamma\delta$).

Clearly, $(x, u, v) \overset{\nu}{\leadsto} (x', u', v')$ if and only if there is a direct path P from x to x' such that $|P|(u \otimes v) = u' \otimes v'$. This is just a reformulation of the execution.

Let ν be a reduced net with n free ports $x_1, \ldots, x_n$. By proposition 2, one sees that there is a finite number of direct paths in ν. So it makes sense to consider the sum $a_{i,j}$ of all $|P|$ where P is a direct path from x_j to x_i. This defines an $n \times n$ matrix A with coefficients in $\Re \otimes \Re$. Consider for instance the following net:



There are two direct paths from x_1 to x_2, and vice-versa. The matrix of this net is:

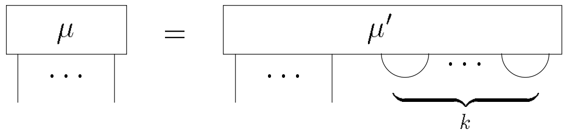
$$A = \begin{pmatrix} 0 & \mathbf{p} \otimes \mathbf{q}^* + \mathbf{q} \otimes \mathbf{p}^* \\ \mathbf{q}^* \otimes \mathbf{q} + \mathbf{p}^* \otimes \mathbf{p} & 0 \end{pmatrix}$$

The matrix of a reduced net will always satisfy the following symmetry property: $a_{j,i} = \phi(a_{i,j})$ where ϕ is the anti-automorphism of $\Re \otimes \Re$ defined by

$$\begin{array}{ll} \phi(\mathbf{p} \otimes 1) = \mathbf{q}^* \otimes 1, & \phi(\mathbf{q} \otimes 1) = \mathbf{p}^* \otimes 1, \\ \phi(1 \otimes \mathbf{p}) = 1 \otimes \mathbf{p}^*, & \phi(1 \otimes \mathbf{q}) = 1 \otimes \mathbf{q}^*. \end{array}$$

Note also that two reduced nets have the same matrix if and only if they are equivalent.

If μ is a net with n free ports, k cuts, and no vicious circle, it can be decomposed as follows:



where μ' is a reduced net with $n + 2k$ free ports. If μ reduces to a reduced net ν, then the matrix A of ν can be obtained by means of the *execution formula*:

$$\begin{array}{rcl} A & = & {}^tJ(A' + A'KA' + A'KA'KA' + \cdots)J \\ & = & {}^tJA'(I - KA')^{-1}J \end{array}$$

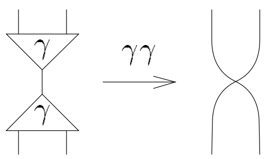
where

- A' is the matrix of μ',
- I is the matrix of the identity on $\{1, \ldots, n + 2k\}$,
- J is the matrix of the inclusion of $\{1, \ldots, n\}$ into $\{1, \ldots, n + 2k\}$,
- K is the matrix of the partial permutation which is undefined on $\{1, \ldots, n\}$ and which exchanges the elements of $\{n + 1, \ldots, n + 2k\}$ pairwise.

The execution formula makes sense because, in that case, the matrix KA' is nilpotent.

3.4 Directed combinators

Our execution is revertible, but the backward execution does not follow exactly the same rules as the forward execution: the γ-stack u must be replaced by $\overline{u}$. This is also expressed by the fact that the anti-automorphism ϕ above maps $\mathbf{p} \otimes 1$ to $\mathbf{q}^* \otimes 1$ instead of $\mathbf{p}^* \otimes 1$. Of course, this would not happen if the rule $\gamma\gamma$ was the same as $\delta\delta$:



This modified system of combinators is not universal in the technical sense of theorem 1, but we shall see that it has the same expressive power as the original one.

To show this, it is natural to consider a system of *directed combinators* with 6 symbols γ, γ^*, δ, δ^*, ε, ε^*, and the 9 interaction rules of figure 16. This system looks very much like the original one, except that it has no symbol which interacts with itself. Because of this, the choice for the annihilations ($\gamma\gamma^*$ and $\delta\delta^*$) is just a matter of taste.

If ν is a net of combinators with n free ports, one defines a net $\tilde{\nu}$ of directed combinators (the *covering* of ν) with $2n$ free ports labelled by $+$ or $-$. The covering is defined on cells as in figure 17. It extends to nets in the obvious way, with the convention that a $+$ must always be plugged into a $-$, as in figure 18.

In fact, we have just separated the forward and the backward executions. This construction is inspired by a classical one in topology: the 2-fold covering of an arbitrary surface by an orientable one.

This covering is obviously not a translation in the technical sense of section 1.7, but it is compatible with reduction:

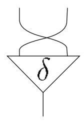
Proposition 7 *If μ reduces to ν in n steps, then $\tilde{\mu}$ reduces to $\tilde{\nu}$ in $2n$ steps.*

Proof: It suffices to check that each interaction between combinators is interpreted by two interactions between directed combinators. **Q.e.d.**

This kind of generalized translation can be called a *2-translation*. A consequence of theorem 1 and proposition 7 is that any interaction system can be 2-translated into the system of directed combinators. Conversely, there is an obvious translation Φ of directed combinators into combinators:

$$\begin{array}{lll} \Phi(\gamma) = \gamma, & \Phi(\delta) = \delta, & \Phi(\varepsilon) = \varepsilon, \\ \Phi(\gamma^*) = \gamma, & \Phi(\delta^*) = \overline{\delta}, & \Phi(\varepsilon^*) = \varepsilon, \end{array}$$

where $\overline{\delta}$ stands for δ with exchanged auxiliary ports:



There is also a translation Ψ of directed combinators into the modified system of combinators, where the rule $\gamma\gamma$ is the same as $\delta\delta$:

$$\begin{array}{lll} \Psi(\gamma) = \gamma, & \Psi(\delta) = \delta, & \Psi(\varepsilon) = \varepsilon, \\ \Psi(\gamma^*) = \overline{\gamma}, & \Psi(\delta^*) = \overline{\delta}, & \Psi(\varepsilon^*) = \varepsilon. \end{array}$$

In particular, this shows that the modified system of combinators has the same expressive power as the original one.

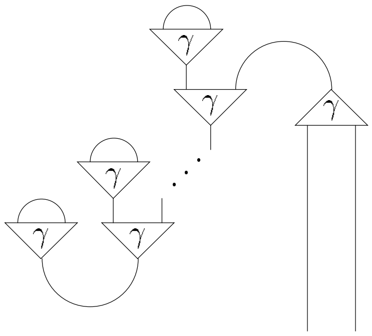
The system of directed combinators itself has some interesting properties. First, it is an extension of the system of multiplicative proof-nets (see [Laf95]). Furthermore, it has a lot of automorphisms which operate on the set

$$\{\gamma, \overline{\gamma}, \gamma^*, \overline{\gamma}^*, \delta, \overline{\delta}, \delta^*, \overline{\delta}^*, \varepsilon, \varepsilon^*\}.$$

One can show that the group of automorphisms has order 16 and is isomorphic to the group of symmetries of a rectangle parallelepiped with square basis (figure 19).

4 Discussion

If a net ν reduces to a wiring ω, one may wonder whether the execution is an efficient algorithm for computing ω. This question was already raised by Vincent Danos and Laurent Regnier in [DR95]. They considered a linear λ-term which corresponds to the following net:



Clearly, such a net reduces to a single wire in a linear number of steps, whereas one can show that the execution takes exponential times! Therefore, the practical interest of this execution is not clear, even if one ignores the parallel aspects of the reduction.

One may also wonder whether our translation of interaction systems into the system of combinators can be seen as a process of compilation. This would mean that we have a physical machine suitable for the reduction of nets of combinators. The problem is that, even if the rules are simple, the geometry of the net may become very complicated during the computation. In fact, we only know that the reduction of interaction nets can be easily and efficiently implemented on a traditional sequential machine, but from this viewpoint, the translation into the system of combinators does not seem to be useful.

From a more theoretical viewpoint, one may also wonder whether there are typed versions of our various systems of combinators. In other words, is there a *logic* behind the interaction combinators ? A simple solution has been given by the author, and independently by Peter Selinger (private communication), but this would lead us too far away from our issue, and we prefer to keep this for subsequent publication.

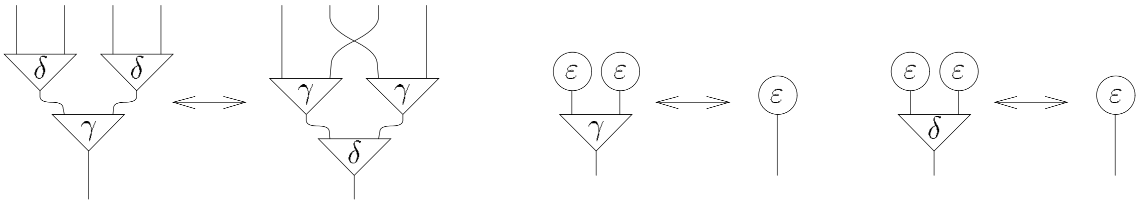


Figure 14: rules for equivalence

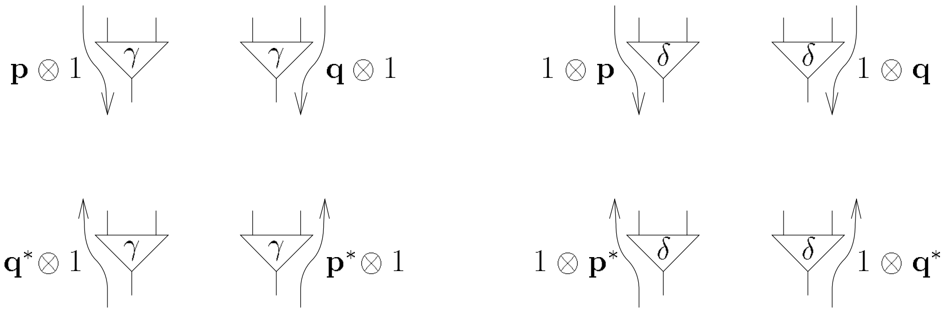


Figure 15: algebraic interpretation

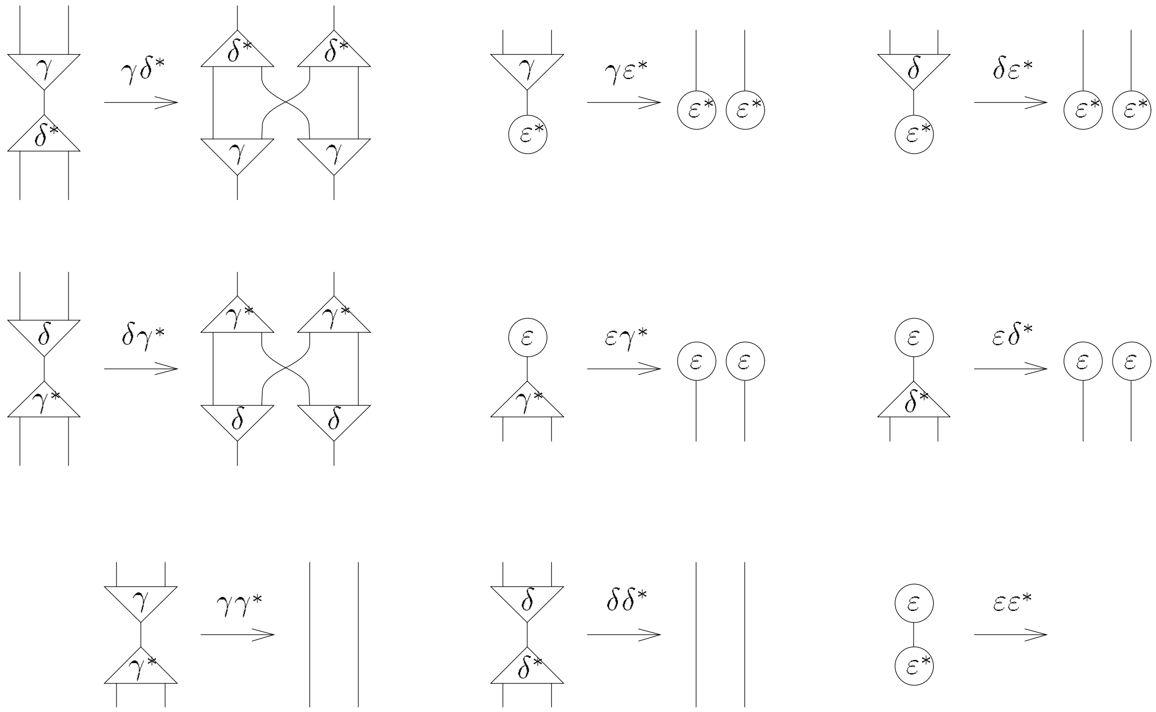


Figure 16: interaction rules for the directed combinators

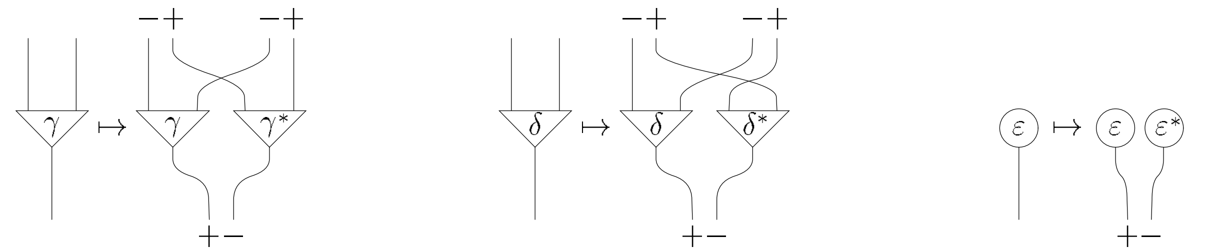


Figure 17: covering of a cell

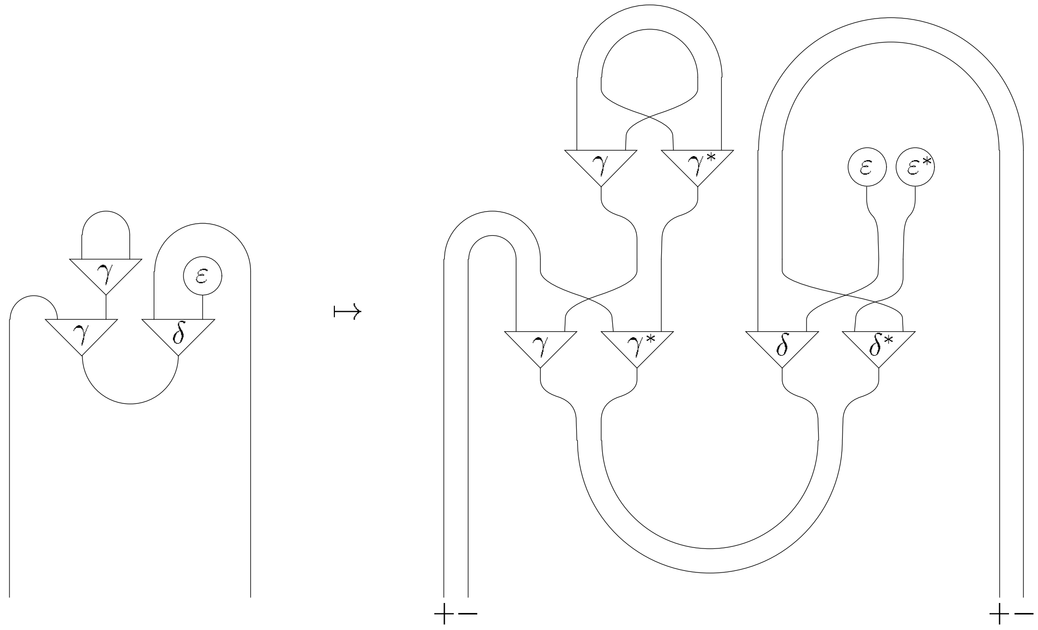


Figure 18: covering of a net

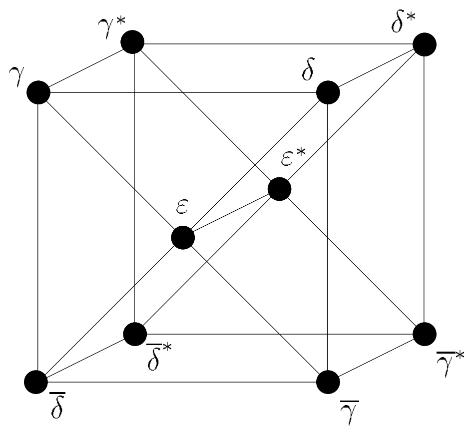


Figure 19: symmetries of the directed combinators

References

[Baw86] A. Bawden (1986), Connection Graphs. In *Proceedings of ACM Conference on Lisp and Functional Programming*, pp 258–265.

[DR95] V. Danos & L. Regnier (1995), Proof-nets and the Hilbert space. In *Advances in Linear Logic* (J.-Y. Girard, Y. Lafont & L. Regnier, editors), London Mathematical Society Lecture Note Series 222, pp 307–328, Cambridge University Press.

[Gay95] S. J. Gay (1995), Combinators for Interaction Nets. In *Proceedings of the Second Imperial College Department of Computing Workshop on Theory and Formal Methods* (C. L. Hankin, I. C. Mackie & R. Nagarajan, editors), Imperial College Press.

[Gir89] J.-Y. Girard (1989), Geometry of Interaction I: interpretation of system F. In *Proceedings Logic Colloquium '88* (Ferro & al., editors), pp 221–260, North Holland.

[Gir95] J.-Y. Girard (1995), Linear Logic: its syntax and semantics. In *Advances in Linear Logic* (J.-Y. Girard, Y. Lafont & L. Regnier, editors), London Mathematical Society Lecture Note Series 222, pp 1–42, Cambridge University Press.

[GAL92a] G. Gonthier, M. Abadi & J.-J. Levy (1992), The geometry of optimal lambda reduction. In *Proceedings of 19th ACM Symposium on Principles of Programming Languages (POPL '92)*, ACM.

[GAL92b] G. Gonthier, M. Abadi & J.-J. Levy (1992), Linear Logic without boxes. In *Proceedings of 7th Annual Symposium on Logic in Computer Science (LICS '92)*, IEEE.

[Laf90] Y. Lafont (1990), Interaction Nets. In *Proceedings of 17th ACM Symposium on Principles of Programming Languages (POPL '90)*, pp 95–108, ACM.

[Laf95] Y. Lafont (1995), From Proof-Nets to Interaction Nets. In *Advances in Linear Logic* (J.-Y. Girard, Y. Lafont & L. Regnier, editors), London Mathematical Society Lecture Note Series 222, pp 225–247, Cambridge University Press.

[Lam90] J. Lamping (1990), An Algorithm for Optimal Lambda Calculus Reduction. In *Proceedings of 17th ACM Symposium on Principles of Programming Languages (POPL '90)*, pp 16–46, ACM.

[Mac94] I. Mackie (1994), *The Geometry of Implementation (an investigation into using the Geometry of Interaction for language implementation)*. PhD thesis, Imperial College of Science Technology and Medicine, London.