Interaction Nets:
Semantics and Concurrent Extensions

PhD Thesis

Damiano Mazza

5th October 2006

Contents

Introduction

In a sense, computer science is as ancient as mathematics; some of its concepts, like the idea of a general procedure to solve a mathematical problem, or the very concept of *computation*, are indispensable to the development of the most elementary ideas of mathematics. It is also certainly older than any device we may nowadays think about when we hear the word "computer", something which has the risk of rendering its English name a little unfit to describe the full scope of this science.

While perhaps the equivalents used in most other European languages, all cognates to the English word *information*, are better behaved in this respect, the essence remains the same: computer science has been and is a fundamental branch of mathematics, even more so as an ever growing host of other areas of human knowledge, from physics to biology, from philosophy to cognitive sciences, appear to develop more and more profound relationships to it. Not to forget, speaking of "computers", about new technologies, which tend to become more and more connected to the advances —not just practical but *theoretical*— made in this science.

Computer science took its modern attire during the third decade of the last century. Using mostly ideas coming from mathematical logic, it started roughly as an attempt to rigorously address questions such as "what does *computing* exactly mean?", and "what does it take to do it?". Today we know that there exists a multitude of different answers, each of these corresponding to a particular *computational model*: Turing machines, the λ-calculus, and partial recursive functions are the very first examples of such models, while more recent proposals include cellular automata, process algebras, and quantum-based models. Of course also the fundamental questions driving computer science have evolved through the years (a notable example being the theory of computational complexity), but the exploration of different models of computation and the search for finer, more abstract mathematical descriptions of them still continue at present.

We have said that the original foundations of computer science lay in mathematical logic; the link between these two disciplines has actually grown stronger over time. In this respect, the work we are now introducing would not even have existed if it were not for at least two achievements which have renewed and refreshened such a connection through the years: Haskel Curry and William Howard's two-step discovery, in 1958 and 1969 respectively, of the correspondence between λ-calculus and natural deduction which bears their names, and Jean-Yves Girard's introduction of *linear logic*, just about twenty years ago.

Interaction nets. It is precisely from the offspring of linear logic that the computational model we are interested in comes from. Interaction nets, introduced by Yves Lafont in 1990, are in fact a generalization of the *proof-nets* of multiplicative linear logic. Proof-nets are graph-like structures which allow to introduce a certain degree of parallelism in the representation of proofs. A typical (and fundamental) example of such parallelism is the associativity of deduction: if we have three lemmas stating respectively that A implies B, B implies C, and C implies D, no reasonable mathematician will ever feel that we are in posses of two *different* proofs that A implies D just because we can compose our lemmas in two different ways. Yet, in the traditional proof-theoretical formalism used for linear logic —sequent calculus—, these two ways correspond to two different objects; on the contrary, if we are composing proof-nets, the result is unique.

In proof-nets, cut-elimination becomes a graph-rewriting process; via the Curry-Howard isomorphism, this allows us to see computation itself, i.e., executions of programs, as a form of graph-rewriting. Such rewriting process is particularly well behaved when we restrict to the so-called multiplicative fragment of linear logic. But the programs one can express in multiplicative linear logic are really too few; the idea behind interaction nets is therefore to generalize the kind of dynamics at work in multiplicative proof-nets so as to obtain systems which are equally well behaved but far more expressive.

As a computational model, interaction nets can be compared to the λ-calculus, which is also a rewriting system. There are indeed many similarities between the two, most notably the existence of a typed and an untyped framework, and the ability to define higher order functions, which is typical of functional programming. But there are also differences: computation in interaction nets is *asynchronous*, i.e., the synchronization between components of a computational process is completely local. On the contrary, the λ-calculus supposes the presence of a global synchronization mechanism for duplicating and erasing arbitrarily large structures. In this respect, interaction nets are not far from being "parallel Turing machines": computational steps are elementary enough to be considered executable in constant time, and several steps may be performed in parallel. This is very interesting from the point of view of computational complexity.

There are also several nice applications of interaction nets. The first one, and most notable, is the implementation of Lamping's algorithm for optimal reduction in the λ-calculus, which is done precisely by using particular interaction nets called *sharing graphs*. Other interaction-net-based implementations or conceptions of functional languages have later been proposed, and there is still quite some work going on in this direction.

Actually, such applicative aspects of interaction nets have largely been preferred in the literature to the theoretical ones. One of the objectives of this thesis is to "restore the equilibrium": we attempt to make an in-depth study of the semantical aspects of interaction nets, in particular finding results comparable to those available for the λ-calculus.

Obsevational equivalence. We start by defining a notion of *observational equivalence* for interaction nets, which will be the basis of all of our further semantical investigations. In the context of a programming language, of which

interaction nets and the λ-calculus are abstract examples, two programs are observationally equivalent when there is no way to tell within the language itself whether one of these programs has been replaced by the other, i.e., they "behave" exactly in the same way. By "within the language itself" we mean that to observe the behavior of a program we are only allowed to act as if we were ourselves programs: we cannot look at the source code of two programs to tell whether they are different; we can instead give the same input to both and read the outcome. In other words, observational equivalence is an interactive notion.

Denotational semantics. After a good notion of observational equivalence is found, we go on to define a *denotational semantics* for interaction nets. Actually, since interaction nets are a paradigm rather than a fixed formal system, we have to choose a particular interaction nets system as the object of our analysis. For this, the system of the *interaction combinators*, also introduced by Yves Lafont, seems to be the best candidate, because of its *universality*: any interaction nets system can be translated into the interaction combinators. Therefore, our denotational semantics will interpret nets of interaction combinators, or rather of *symmetric combinators*, a technical variant which is easier to model.

In general, the goal of denotational semantics is to give a description of programs which is more abstract than that given by the syntax, and which thus lends itself better to mathematical analyses and proof techniques. This is supposed to allow one to prove properties about programs which would be otherwise impossible, or very difficult to prove using only syntactical tools. In our context, we would like our mathematical description to capture the essence of the interactive behavior of our programs, which we have seen above is the object of observational equivalence. Hence, to us, *the ultimate goal of denotational semantics is to find an abstract mathematical structure in which observational equivalence becomes an equality.* A denotational semantics verifying this is usually said to be *fully abstract*.

Unfortunately, we shall only partially fulfill this objective. In fact, we have not been able to prove a full abstraction result for the class of models we shall define; we are able though to exhibit other structures, similar to *Böhm trees* in the λ-calculus, which characterize the interactive behavior of nets, and which may be the basis for finding a "more semantical", fully abstract model.

Geometry of interaction. Another component of our semantical study of the symmetric combinators is the *geometry of interaction* (GoI). Introduced by Girard in the context of linear logic, the geometry of interaction has the objective of giving a more "operational" semantics of computation, in which the execution of a program receives an elaborate mathematical meaning. In its original formulation, the GoI semantics interprets linear logic proofs as pairs of operators on the (denumerably) infinite-dimensional Hilbert space, only one of which contains information in the case of a cut-free proof. The cut-elimination process is interpreted as a way of mathematically composing these two operators to obtain a third one, which is the interpretation of the cut-free result.

After Girard's one, more "down-to-earth" GoI interpretations have been proposed; in this thesis, we develop the GoI semantics sketched by Lafont in his work introducing the interaction combinators. We interpret nets of (symmetric)

combinators as homomorphisms of certain monoids, called *interaction monoids*, and we give the mathematical formulation of execution in the terms described above. These interaction monoids can also be used to interpret nets according to the denotational semantics previously introduced; in fact, we show that there actually exists a nice relation between the denotational and the GoI semantics.

Concurrency. We said above that interaction nets are a model of *distributed* computation: the execution of a program takes place at several different locations, without any global synchronization. This implies the presence of a certain *parallelism*, and indeed at some point we made the analogy of interaction nets as "parallel Turing machines". Yet, the parallelism at work in interaction nets has a peculiar property, namely that *no conflict can ever arise among components of a computational process.* In fact, the structure of interaction nets is such that, for example, two threads of a program will never compete to access the same resource.

This particular form of parallelism guarantees an exceptional property of interaction nets, which is their strong determinism: not only the result of a computation, but the computation itself is essentially unique. This is another remarkable difference with respect to the λ-calculus: reduction strategies are meaningless in interaction nets, at least from the point of view of efficiency.

However, this very nice property becomes an extremely heavy limitation if we want to model realities in which conflicts not only arise, but are actually the central feature we want to express. This is the case of concurrency theory, a fairly recently developed but rapidly growing field of utter theoretical and practical interest these days.

The rise of concurrency theory is a response to the ever growing importance of computer networks and mobile technologies, which challenge the descriptions of computational process given by the traditional models mentioned above, such as Turing machines and λ-calculus. In modern computer networks, conflicts like those suggested above arise all the time: two workstations wanting to use the same printer at the same time, two customers wanting to book the same flight at the same time, etc. A correct development of such networks, and of softwares dealing with such realities requires a solid theory, just as the older results of theoretical computer science have previously been essential for the progress of sequential machines and softwares.

Many of the computational models introduced for concurrency theory gather under the name of *process calculi*. A particularly successful one from a theoretical viewpoint is Robin Milner's *π-calculus*, which, in one form or another, is used by many as the standard model for concurrency theory. It is not as "stable" as the λ-calculus, but there is a vast number of results and proof techniques available for it, which can be used as a source of inspiration in the definition and development of other concurrent models.

As we said, interaction nets are doomed right from the start if we want to use them as a model of concurrent computation. This is precisely the second objective of our thesis: to define a concurrent extension of interaction nets, and to adapt to it as many results existing on interaction nets as possible.

Multiport interaction nets. The extension we introduce here, called *multiport interaction nets*, is a quite natural one, and we found out that it had

actually already been considered in two other Ph.D. thesis, Vladimir Alexiev's (1999) and Lionel Khalil's (2003). It is a conservative extension, i.e., interaction nets are special cases of multiport interaction nets.

The first thing we verify is the expressive power of such extension, and prove that it is just as expressive as the π-calculus. As a by-product, we obtain a graphical encoding for the π-calculus, which in many respects works better than the previously existing ones: it covers all of the features of the calculus (even though the encoding becomes heavily technical in presence of guarded choice, which by the way is not considered to be fundamental in the π-calculus), and it decomposes process reduction into purely local and asynchronous elementary steps, much like the above-mentioned sharing graphs for the λ-calculus. At the same time, the encoding seems to be easily adapted to a higher-order framework, as this is actually a quite natural setting for interaction nets.

The next step would be to develop for multiport interaction nets the notions introduced in the deterministic case, namely behavioral equivalence and denotational semantics. This is unfortunately a little too ambitious to be realized here; this thesis contains just the first results going in this direction.

Universality. As remarked above, when defining a denotational semantics for interaction nets one must restrict to a particular system, possibly an interesting one. The interaction combinators are such a system, because by universality any denotational semantics for them is also, via an encoding, a denotational semantics for any other interaction nets system. Therefore, the task of finding a denotational semantics in the multiport case cannot be properly addressed if we do not first find the equivalent of the interaction combinators in this new setting.

By definition, an interaction nets system is universal just if any other interaction nets system can be translated into it. Thus, the key concept concerning universality is the definition of *translation*. In the deterministic case, Lafont gave what is arguably the most natural definition. In the multiport case, the situation is rendered delicate by the presence of non-determinism, and a good notion of translation is harder to define. We propose one which makes use of behavioral equivalence: the translation of a net must be able to simulate such net up to some congruence. The stronger this congruence, the more convincing the translation will be.

The multiport combinators. This brings up the issue of defining a behavioral equivalence for the proposed universal system, a task which we are not able to fulfill in general. However, we do have a partial, but fairly satisfactory result. We define a system, called the *multiport combinators*, and we define a good notion of behavioral equivalence for a small subsystem of it. Then, we exhibit a map from any multiport interaction nets system to the multiport combinators. This map has the property that whenever a net μ evolves to a net μ' in the system we want to translate, then the image of μ evolves to the image of μ' in the multiport combinators. We are able to prove that this map is a translation up to the behavioral equivalence defined for the above-mentioned subsystem, and thus only for such subsystem. Actually, we conjecture that it is a translation in all cases, but we are lacking a general notion of behavioral equivalence to prove it. The result is nevertheless satisfactory because the subsystem in question is

as expressive as a relatively large fragment of the π-calculus (roughly speaking, only guarded choice is not covered).

The overall structure of our work is divided into two parts, corresponding to the two main objectives of the thesis we have just described. We shall now enter into the details of each of them.

Semantics of Interaction Nets

Interaction Nets

This is an introductory chapter, containing the main definitions about interaction nets, fundamental to the rest of the work. The main references for this chapter are [Laf90, Laf97, Laf95].

Cells, wires, nets. In Sect. 1.1 we define the basic components of interaction nets, namely *cells* and *wires*: the first are "agents" characterized by a symbol and a certain number of *ports*, of which exactly one must be *principal*, the others being *auxiliary*; the second are used to plug the ports of cells together, to form *interaction nets*. The introduction is done in two parts: we first give a formal definition of interaction net, and then present the less formal graphical representations used in practice. We also define important configurations used in the sequel, such as *active pairs* (two cells with a wire connecting their principal ports) and *vicious circles* (sort of cycles of principal ports, corresponding to deadlocks), and define *cut-free nets* as nets containing no active pair and no vicious circle.

Interaction rules: β-equivalence and totality. In Sect. 1.2 we introduce *interaction rules*, which are the graph-rewriting rules driving the computational process in interaction nets. The key property of interaction nets is that the left member of interaction rules must be an active pair, and that each active pair has at most one interaction rule. Whenever a net μ can be rewritten to μ' by means of one such rule, we write $\mu \to \mu'$; we define $\to^*$ as the reflexive-transitive closure of $\to$, and we define β-equivalence as usual: $\mu \simeq_\beta \mu'$ iff there exists μ'' such that $\mu \to^* \mu''$ and $\mu' \to^* \mu''$. We say that a net μ is *total* iff there exists a cut-free net ν such that $\mu \to^* \nu$. Reduction is strongly confluent: if $\mu \to \mu_1$ and $\mu \to \mu_2$, then there exists μ' such that $\mu_1 \to \mu'$ and $\mu_2 \to \mu'$. This implies that $\simeq_\beta$ is an equivalence relation, that the cut-free form is unique if it exists, and that also the reduction of a net is unique, up to trivial permutations of rules. We also give the definition of *interaction nets system* (INS), and introduce typing and deadlock-freeness, together with the associated correctness criterion. These two notions are mentioned only for the sake of self-containedness; we shall never make use of them, except in a few examples.

Examples: proof-nets, interaction combinators, partial recursive functions. Sect. 1.3 is dedicated precisely to giving detailed examples of interaction nets systems. Such examples are chosen for their usefulness in the sequel, or because they complement existing examples found in the literature. The first one is the source of inspiration for the whole paradigm: multiplicative

linear logic proof-nets. Then, we immediately introduce the interaction combinators and their symmetric variant, the latter being essential in the sequel. We also give an explicit proof of the Turing-completeness of the symmetric combinators by encoding the **SK** combinators in them, an exercise which we have not seen done in any previous work. We conclude with an INS capable of encoding all partial recursive functions, probably the only classical computational model which, to our knowledge, had not been translated into interaction nets.

Universality. Sect. 1.4 deals with the *universality* of the interaction combinators. The notion of translation between interaction nets systems is defined; it requires a system to be able to simulate another not just in expressiveness but also as far as its complexity of reduction and degree of parallelism are concerned. A universal system is thus defined as an INS in which all other INS's can be translated. The interaction combinators are universal; the symmetric combinators are not, although virtually any "interesting" INS can be translated into them (Theorem 1.9). We close with a few (new, as far as we know) remarks on a subsystem of the interaction combinators (symmetric or not).

Observational Equivalence

This chapter opens the true original content of the thesis. To our knowledge, the only existing pieces of work dealing with similar matters are Lafont's paper introducing the interaction combinators [Laf97], and Maribel Fernández and Ian Mackie's work on operational equivalence [FM03].

Path-based observational equivalence. In Sect. 2.1 we define the observational equivalence we use throughout the rest of the first part of the thesis. The key notion is that of *observable path*. The importance of paths in proof-nets (and interaction nets) has been pointed out by more than fifteen years' worth of work on the geometry of interaction; in fact, the paths we define for our nets, called *interaction paths*, are an extension of Danos&Regnier's *straight paths* [DR95] (the two notions indeed coincide in the interaction combinators, symmetric or not).

Observable paths are simply interaction paths entering and exiting a net through two of its free ports, and which are stable under reduction. We write $\mu\downarrow$ if a net μ has an observable path; of course, by definition $\mu\downarrow$ and $\mu \to^* \mu'$ implies $\mu'\downarrow$. We say that a net μ is *observable* iff $\mu \to^* \mu'\downarrow$, and we write $\mu\Downarrow$. If μ reduces to no observable net, we say that it is *blind*, and we write $\mu\Uparrow$.

We then introduce the notion of *context*, which is particularly natural in interaction nets: a context C for nets with n free ports is a net with at least $n+1$ free ports, and if μ is a net with n free ports, the application of C to μ, written $C[\mu]$, is the net obtained by simply plugging all free ports of μ to some of the free ports of C (the information concerning which ports must be used must therefore come together with the net C in the proper definition of context).

Our observational equivalence is defined as follows: two nets μ, ν with the same number of free ports are observationally equivalent, and we write $\mu \simeq \nu$, iff $C[\mu]\Downarrow \Leftrightarrow C[\nu]\Downarrow$ for every context C. In other words, any context either makes both μ and ν observable, or both blind. As expected in a deterministic framework such as interaction nets, we have $\simeq_\beta \subseteq \simeq$.

There are strong reasons to think that observability and blindness correspond to the λ-calculus notions of solvability and unsolvability, and that therefore $\simeq$ corresponds to head-normal-form equivalence in the λ-calculus (two λ-terms T, U are hnf-equivalent iff, for every context C, $C[T]$ and $C[U]$ are either both solvable or both unsolvable). This is explained in Sect. 2.1.3, right after the introduction of $\simeq$. Another interesting thing is the comparison to Fernández&Mackie's definition of observational equivalence, very briefly discussed at the end of the same section.

Bisimilarity. Sect. 2.2 redefines the observational equivalence introduced above by means of a notion of *bisimilarity*. Two nets μ, ν are bisimilar, written $\mu \mathrel{\dot{\sim}} \nu$, iff $\mu\downarrow$ implies $\nu\Downarrow$, $\mu \to \mu'$ implies $\nu \to^* \mathrel{\dot{\sim}} \mu'$, and the same holds with the rôles of μ and ν exchanged. The determinism of interaction nets trivializes the definition of bisimilarity, and in fact we prove that $\mu \mathrel{\dot{\sim}} \nu$ iff μ and ν are either both observable or both blind. But this is actually quite interesting for us, because it implies that observational equivalence can be redefined as $\mu \simeq \nu$ iff $C[\mu] \mathrel{\dot{\sim}} C[\nu]$ for every context C, i.e., observational equivalence is bisimilarity under any context (this is what is usually called *barbed congruence* in the π-calculus and similar process calculi). This opens up the possibility of proving observational equivalence by means of coinduction; the rest of the section is in fact dedicated to the development and application of such typical proof techniques, used later on in the thesis.

Interaction combinators and internal separation. Sect. 2.3 applies the previous two sections to the fundamental case of the interaction combinators. It contains one of the main results of our work, the Separation Theorem 2.19, which is analogous to Böhm's Theorem for the λ-calculus. First of all, we introduce further rewriting rules for the interaction combinators, which induce an equational theory on nets denoted by $\simeq_\eta$. These rules are not interaction rules, but, as the notation suggests, they play the same rôle as the η-rule in the λ-calculus. We define $\beta\eta$-equivalence as the transitive closure of $\simeq_\beta \cup \simeq_\eta$, and, using the coinductive techniques introduced above, we prove that $\mu \simeq_{\beta\eta} \nu$ implies $\mu \simeq \nu$.

In Sect. 2.3.3 we prove the Separation Theorem; one of its consequences is that, if we restrict to total nets, then the above implication can be reversed. Actually, the result is stronger: we consider two nets W and E, which are the cut-free "incarnations" of resp. the observable and of the blind net, and we state that, given two total nets with the same number of free ports such that $\mu \not\simeq_{\beta\eta} \nu$, then there exists a context C (which moreover has a particularly simple form) such that $C[\mu] \to^* W$ and $C[\nu] \to^* E$, or viceversa. This was indeed our starting point to define the observational equivalence $\simeq$.

An immediate application of the Separation Theorem is the maximality of $\simeq_{\beta\eta}$ on total nets: if $\sim$ is a non-trivial congruence on total nets containing $\simeq_\beta$, then $\sim \subseteq \simeq_{\beta\eta}$. We conclude by giving some remarks on internal separation, and by introducing η- and $\beta\eta$-equivalence for the symmetric combinators, mentioning that the Separation Theorem holds also for this system.

Totality-base observational equivalence. In Sect. 2.4 we define an alternative observational equivalence, based on totality rather than observability:

$\mu \simeq^\circ \nu$ iff, for every context C, either $C[\mu]$ and $C[\nu]$ are both total, or they are both non-total. While this equivalence seems to correspond to normal-form equivalence in the λ-calculus (two λ-terms T, U are nf-equivalent iff, for every context C, $C[T]$ and $C[U]$ are either both normalizable or both non-normalizable), it is readily seen that this is not the case. In fact, nf-equivalence strictly implies hnf-equivalence in the λ-calculus, whereas in interaction nets $\simeq^\circ$ and $\simeq$ are orthogonal, i.e., neither is included in the other. This is because totality does not coincide with normalizability; it is stronger than it.

Nevertheless, $\simeq^\circ$ does coincide with $\simeq$ (and thus with $\simeq_{\beta\eta}$) on total nets, and in the case of the interaction combinators (symmetric or not) it accepts a definition closely resembling the definition of observational equivalence in Girard's ludics [Gir01]. This allows a nice comparison between syntactical and topological separation, which is done in Sect. 2.4.3.

The Symmetric Combinators

This chapter studies to a certain depth the symmetric combinators, defining several denotational semantics for them, and analyzing the properties of each. The only other work attempting a semantical study of the interaction combinators (symmetric or not) is Lafont's paper introducing them [Laf97], in which he sketched the GoI semantics we develop here.

Denotational semantics. Sect. 3.1 starts with the definition of a very simple class of denotational models for the symmetric combinators, inspired by the relational semantics of linear logic. In the multiplicative case, the relational semantics interprets a formula A by a set $|A|$, and a proof of A by a subset of $|A|$. The two multiplicative connectives are both interpreted by the Cartesian product: $|A \otimes B| = |A \parr B| = |A| \times |B|$.

The symmetric combinators have three kinds of cells, two with two auxiliary ports and one with zero. If they are of the same kind, two binary cells interact with each other in much the same way a tensor and a par interact in proof-nets; if they are of different kind, they "commute", duplicating each other. The idea is then to consider the relational interpretation, but of course untyped; we take an infinite pointed set $\mathcal{D}$, the distinguished element of which we denote by $\mathbf{0}$, and we interpret each binary cell using a bijection between $\mathcal{D} \times \mathcal{D}$ and $\mathcal{D}$, and the zeroary cell with $\mathbf{0}$. If we denote by $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$ these two bijections, to correctly interpret the interaction rules we need the following equalities to hold:

$$\langle \mathbf{0}, \mathbf{0} \rangle = [\mathbf{0}, \mathbf{0}] = \mathbf{0}$$

$$\langle [a, b], [c, d] \rangle = [\langle a, c \rangle, \langle b, d \rangle], \text{ for all } a, b, c, d \in \mathcal{D}$$

An infinite pointed set satisfying all of the above is called an *interaction set*.

Two bijections verifying the second equation (which is reminiscent of the "exchange rule" for the composition of 2-arrows in a 2-category) are said to be *companions*. In Sect. 3.1.1 we make sure that companion bijections do exist, and immediately find examples of interaction sets. The simplest example is $\mathbb{N}^*$, the set of strictly positive integers. By the fundamental theorem of arithmetics, each positive integer n admits a unique decomposition as a product of powers of primes:

$$n = \prod p_i^{h_i}$$

where p_i is the sequence of prime numbers, and h_i is an almost-everywhere zero sequence of non-negative integers. Therefore, we define two functions from $\mathbb{N}^* \times \mathbb{N}^*$ to $\mathbb{N}^*$ as follows:

$$\langle \prod p_i^{h_i}, \prod p_i^{h_i} \rangle = \prod p_{2i}^{h_i} \prod p_{2i+1}^{k_i}$$

$$[\prod p_i^{h_i}, \prod p_i^{h_i}] = \prod p_i^{\beta(h_i,k_i)}$$

where β is a bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$ such that $\beta(0,0) = 0$. These two functions can in fact be checked to be companion bijections.

The interpretation of a net in an interaction set is defined by means of *experiments*. Introduced by Girard for linear logic [Gir87a], experiments allow one to compute the interpretation of a proof-net without looking at one of its corresponding sequent calculus proofs. In other words, experiments allow the definition of a denotational interpretation on objects which are not necessarily built inductively, as in our case.

An experiment e on a net μ in an interaction set $\mathcal{D}$ is an assignment of elements of $\mathcal{D}$ to the ports of μ, satisfying the following requirements:

- if the two auxiliary ports a binary cell of μ of the first kind are labelled by c and d, then the principal port of the same cell is labelled by $\langle c, d \rangle$;
- the same is true for the second kind of binary cells, but using the bijection $[\cdot, \cdot]$;
- the principal port of a zeroary cell is always labelled by $\mathbf{0}$.

If μ has n free ports, the labels d_i assigned by e to the free ports of μ form a tuple $(d_1, \ldots, d_n)$, which is called the *result* of e, and is denoted by $|e|$.

The denotational interpretation in an interaction set $\mathcal{D}$ of a net μ, denoted by $[\![\mu]\!]$, is the set containing all the results of all experiments on μ in $\mathcal{D}$:

$$[\![\mu]\!] = \{|e| \ ; \ e \text{ experiment on } \mu \text{ in } \mathcal{D}\}.$$

One can show that this defines a denotational semantics for the symmetric combinators: $\mu \simeq_{\beta\eta} \nu$ implies $[\![\mu]\!] = [\![\nu]\!]$ (the interpretation models $\beta\eta$-equivalence), which in turn implies that, for every context C, $[\![C[\mu]]\!] = [\![C[\nu]]\!]$ (the interpretation induces a congruence), and there exist two nets μ, ν such that $[\![\mu]\!] \neq [\![\nu]\!]$ (the interpretation is non-trivial). A simple additional property of interpretations is that they are always pointed, i.e., for all μ, $(\mathbf{0}, \ldots, \mathbf{0}) \in [\![\mu]\!]$.

If we restrict to total nets, this interpretation has two additional properties: it is *injective* with respect to $\simeq_{\beta\eta}$, and there exists a characterization of the subsets of $\mathcal{D} \times \cdots \times \mathcal{D}$ which are the interpretation of some net (a full completeness result). Injectivity is nothing but the converse of the first implication above: if μ, ν are two total nets such that $[\![\mu]\!] = [\![\nu]\!]$, then $\mu \simeq_{\beta\eta} \nu$. This of course does not hold for arbitrary nets; something similar happens in the λ-calculus, where for example Ω and $\lambda x.\Omega$ are not $\beta\eta$-equivalent, and yet receive the same interpretation in any sensible model.

The last important result of the section is a semantical characterization of observability (Corollary 3.24), in a certain interaction set which we call $\Phi_2(\mathcal{A})$. This characterization states that, if we interpret nets in $\Phi_2(\mathcal{A})$, then a net μ is observable iff $[\![\mu]\!] \neq \{(\mathbf{0}, \ldots, \mathbf{0})\}$. Since observability corresponds to solvability,

this result basically tells us that $\Phi_2(\mathcal{A})$ is the equivalent of a sensible model for the λ-calculus.

The "only if" part of the characterization is a straightforward consequence of the definition of observability and of the basic properties of interaction sets; the "if" part is on the contrary quite hard to prove, and requires realizability techniques similar to those introduced by Jean-Louis Krivine [Kri90]. Very briefly, we define what it means for a net μ to *realize* an element $\vec{x}$ of $\Phi_2(\mathcal{A}) \times \cdots \times \Phi_2(\mathcal{A})$, which we denote by $\mu \Vdash \vec{x}$; the set of all nets μ such that $\mu \Vdash \vec{x}$ is called the set of realizers of $\vec{x}$. Then, we prove two fundamental results: the Adaptation Lemma 3.18, stating that the realizers of any $\vec{x} \neq (\mathbf{0}, \ldots, \mathbf{0})$ are all observable nets, and the Adequacy Lemma 3.22, stating that if $\vec{x} \in [\![\mu]\!]$, then $\mu \Vdash \vec{x}$. Put together, these two lemmas (whose proofs rely on a host of rather technical intermediate results) obviously yield the "if" part of the characterization.

We close this section by giving a technical justification to our choice of modeling the *symmetric* combinators instead of the interaction combinators, which enjoy a stronger universality property and which would thus seem more interesting to analyze.

Full abstraction. Sect. 3.2 is an attempt to address the problem of full abstraction for the symmetric combinators. As we said above, in our case a denotational semantics achieves full abstraction iff, for all nets μ, ν, $\mu \simeq \nu$ iff $[\![\mu]\!] = [\![\nu]\!]$. Unfortunately, we do not know at present of any interaction set yielding a fully abstract semantics. The above mentioned characterization of observability trivially implies one of the two parts of full abstraction, namely what is usually called the *adequacy* of a semantics: in $\Phi_2(\mathcal{A})$, $[\![\mu]\!] = [\![\nu]\!]$ implies $\mu \simeq \nu$.

To look for hints on how to obtain full abstraction, we define a quite interesting kind of structure, called *edifice*. An edifice is the symmetric combinators' equivalent of a Böhm tree; however, because of the symmetries of interaction nets, i.e., the fact that no free port can be privileged with respect to the others, edifices are more complex (but more symmetrical) than trees.

An edifice is a set of unordered pairs of infinite words, called *arches*. After defining a distance on arches, which is based on Cantor's distance for infinite binary words, we consider edifices as metric spaces, and deem an edifice to be *complete* iff it is a complete metric space according to the metric defined. Given an edifice $\mathfrak{E}$, we denote by $\overline{\mathfrak{E}}$ its completion.

The first step is to assign to each cut-free net ν a complete edifice $\mathfrak{E}^\bullet(\nu)$, whose arches basically correspond to the observable paths of ν. In order to extend the assignment to arbitrary nets, we define the notion of *approximation*: up to some details, a cut-free net μ_0 approximates μ, which we denote by $\mu_0 \sqsubseteq \mu$, iff μ reduces to a net μ' containing μ_0 and such that, for any reduct μ'' of μ', also μ'' contains μ_0. In other words, μ_0 is a "piece" of the cut-free form of μ (if this exists).

The idea is that an edifice must represent the cut-free form of a net μ even if this is not total, i.e., it must be a sort of "infinite normal form". Then, we define the set of *approximate edifices* of a net μ by $\mathfrak{Apx}(\mu) = \{\mathfrak{E}^\bullet(\mu_0)\ ;\ \mu_0 \sqsubseteq \mu\}$. At this point, the most natural thing to do would be to take the union of the edifices of all approximations of μ to be the edifice of μ; but this edifice is not

complete in general, so we rather set

$$\mathfrak{E}(\mu) = \overline{\bigcup \mathfrak{Apx}(\mu)}.$$

We are then able to prove that interpreting nets as edifices yields a denotational semantics, i.e., $\mu \simeq_{\beta\eta} \nu$ implies $\mathfrak{E}(\mu) = \mathfrak{E}(\nu)$. But this time we can do more (cf. Theorem 3.33): if $\mathfrak{E}(\mu) \neq \mathfrak{E}(\nu)$, then we are able to find a context discriminating between μ and ν, i.e., $\mu \not\simeq \nu$, which is the contrapositive of the full abstraction property.

The proof of this fact heavily relies on the completeness of the edifices interpreting nets. In fact, in Sect. 3.2.3 we shall show an example of a net which reduces forever, and yet is observationally equivalent to a simple wire. This net admits as approximations an infinity of cut-free nets, each "almost" η-equivalent to the wire; the "almost" comes from the fact that one observable path is missing. This observable path actually forms "in the limit", when reduction is carried on forever. Completeness precisely accounts for this fact: it makes of this intuition of "taking the reduction to the limit" a true topological limit.

The situation is in fact very similar to the so-called "infinite η-expansion" of λ-terms, considered by Christopher Wadsworth and Martin Hyland when proving that Scott's D_∞ model is fully abstract with respect to what we called hnf-equivalence. In fact, it is well known that in the λ-calculus one can find a non-normalizable term J which is hnf-equivalent to the identity $I = \lambda z.z$.

Geometry of interaction. Sect. 3.3, the last of Chapter 3, develops the geometry of interaction (GoI) semantics sketched by Lafont for the interaction combinators [Laf97]. We require interaction sets to have a minimal of algebraic structure, and we thus define *interaction monoids*, the distinguished element of the pointed set being the zero of the monoid (which is supposed to be commutative). Then, given an interaction monoid M, we associate to a net with n free ports and k active pairs and/or vicious circles two endomorphisms $\mu^\bullet$ and σ of M^{n+2k}, where $M^i = M \oplus \cdots \oplus M$ i times. The assignment is such that $\sigma = \mathbf{0}$ (the everywhere-zero endomorphism) iff μ is cut-free. Then, Lafont has proved that, if μ is total and if its cut-free form is ν, the GoI interpretation of ν is obtained by means of Girard's *execution formula*:

$$\nu^\bullet = \pi^t \left(\sum_{i=0}^{\infty} \mu^\bullet (\sigma \mu^\bullet)^i \right) \pi,$$

where π is the matrix of the inclusion homomorphism of M^n into M^{n+2k}, and π^t its transposed, i.e., the homomorphism from M^{n+2k} to M^n "chopping off" the last $2k$ components of an element. Composition of two homomorphisms is denoted by juxtaposition, i.e., $\sigma\mu^\bullet = \sigma \circ \mu^\bullet$.

The execution formula makes sense because one can prove that, under the hypotheses that μ is total, $\sigma\mu^\bullet$ is nilpotent. Here we show that the converse holds as well (Theorem 3.43), i.e., we give a characterization of totality by nilpotency: μ is total iff $\sigma\mu^\bullet$ is nilpotent. This is similar to what done by Vincent Danos and Laurent Regnier in the λ-calculus [DR95].

Finally, we show that, in the case of cut-free nets, there exists a nice connection between the GoI semantics and the denotational semantics based on

interaction sets (which can be defined on interaction monoids as well, these being special cases of interaction sets). In fact, if ν is a cut-free net, we have

$$[\![\nu]\!] = \mathrm{fix}(\nu^\bullet),$$

where the interpretation is taken in any interaction monoid M, and $\mathrm{fix}(\nu^\bullet)$ denotes the submonoid of the fixpoints of the endomorphism $\nu^\bullet$, which is the GoI interpretation of ν in M (we remind that, ν being cut-free, we have $\sigma = \mathbf{0}$).

Multiport Interaction Nets and Concurrency

Multiport Interaction Nets

The first two sections of this chapter introduce multiport interaction nets. These are obtained from interaction nets simply by allowing cells to have more than one principal port. The definitions of wire, net, and interaction rule remain the same; in particular, active pairs are still pairs of cells with a wire connecting two of their principal ports, and there still can be at most one rule defined for each active pair.

The fundamental difference is that, in the multiport case, one cell may be involved in several active pairs, as many as its principal ports. The choice of which one is reduced is non-deterministic; in Sect. 4.3 we show an immediate example of how this implies the presence of non-confluent nets. In particular, we show how to implement a random binary stream generator, i.e., a net having as normal forms all finite binary words. We also show how a few parallel algorithms can be implemented, such as *parallel or* and *bottom-avoiding merge*.

As the deterministic case, we also introduce typed systems, but this time we shall make use of them later on (cf. Chapter 5). "Multiport interaction nets system" is abbreviated with mINS. If a system uses cells with at most n principal ports, we say that it is an nINS; if $n = 1$, we are back to usual INS's.

Multiport interaction nets have already been introduced by Vladimir Alexiev in his Ph.D. thesis [Ale99], as one of several non-deterministic extensions of interaction nets. Alexiev's concern was principally the inter-representability of these extensions into one another; he concluded that all of them are equally expressive, except an extension in which the topology of cells and nets is left untouched, but several rules for each active pair are allowed. This extension, which is the one independently defined by Thomas Erhard and Laurent Regnier in their *differential interaction nets* [ER06], is proved to be unable to faithfully encode multiport interaction nets. However, a recent work by Ehrhard himself with Olivier Laurent [EL06] has shown that, in spite of this, it makes perfect sense to say that differential interaction nets can express concurrent computations.

The only other work dealing with multiport interaction nets is Lionel Khalil's Ph.D. thesis [Kha03], in which he shows how the multiport extension can actually be encoded in interaction nets plus McCarthy's amb, which is represented by a cell with two principal ports and two auxiliary ports. Nevertheless, it is often much more practical to use cells with an arbitrary number of principal ports, and this is why we prefer to keep working within this larger framework.

Essentially, our thesis adds two main contributions to the theory of multiport interaction nets: an encoding of the full π-calculus, and a universal system of

multiport combinators, the multiport equivalent of the interaction combinators. These are treated resp. in Chapters 5 and 6.

Encoding the π-calculus

In Sect. 5.1 we briefly recall the basic definitions concerning the π-calculus: processes, structural congruence (denoted as usual by $\equiv$), reduction, and transitions. We also define barbed bisimulation and barbed congruence, the latter usually being considered as the standard behavioral equivalence for the π-calculus. We conclude by defining two subcalculi which will be of interest in the rest of the chapter: the finite π-calculus Fπ, only featuring name passing and name restriction, and the "core" π-calculus Cπ, which also includes replication, and is thus virtually as expressive as the full π-calculus. Neither subcalculus includes guarded choice. The main reference for this section is [SW01].

Encoding the finite π-calculus. Sect. 5.2 introduces a mINS $\mathcal{F}_\infty$ in which Fπ can be faithfully encoded. In order to represent communication channels, the system uses an infinite family of cells with an arbitrarily large number of principal ports; for this reason, we say that $\mathcal{F}_\infty$ is an infinite mINS.

The encoding works as follows. Each process P of Fπ is assigned a net $[P]$ of $\mathcal{F}_\infty$. A single internal transition in Fπ (denoted by $\xrightarrow{\tau}$) is simulated by several reduction steps in $\mathcal{F}_\infty$; however, there is one particular interaction rule which in some sense "marks" the synchronization of two processes. We write $\mu \twoheadrightarrow \mu'$ iff a net μ of $\mathcal{F}_\infty$ reduces to a net μ' by applying such interaction rule exactly once, preceded and followed by the application of any number of other rules.

Because of the decomposition of a single π-calculus transitions in several steps, a generic reduct of a net of the form $[P]$ may not be itself the encoding of a process. For this reason, we introduce a *readback* operation which, given a net μ which is the reduct of a net of the form $[P]$, yields a net $\widetilde{\mu}$ which is of the form $[Q]$ for some process Q.

Then, given any process P of Fπ, the following results can be proved:

- $P \equiv Q$ implies $[P] = [Q]$ (Proposition 5.3);
- if $P \xrightarrow{\tau} Q$, then $[P] \twoheadrightarrow \mu$ such that $\widetilde{\mu} = [Q]$ (completeness, Theorem 5.8).
- if $[P] \twoheadrightarrow \mu$, then $P \xrightarrow{\tau} Q$ such that $[Q] = \widetilde{\mu}$ (soundness, Theorem 5.8).

Adding replication and other features. In Sect. 5.3 we extend the above system, finding a mINS $\mathcal{C}_\infty$ in which Cπ can be encoded. The encoding uses local duplication to implement replication, much like *fan nodes* implement copying in sharing graphs. The same results mentioned above can be proved for this encoding.

The two mINS's introduced so far have an unpleasant feature: they are infinite, i.e., the number of principal ports used in unbounded, and depends on the complexity of the process we are translating, in particular on how many of its subprocesses communicate at the same time on the same channel. In Sect. 5.4 we take care of this problem, showing that $\mathcal{F}_\infty$ and $\mathcal{C}_\infty$ can be encoded resp. by two systems $\mathcal{F}$ and $\mathcal{C}$, both finite. In fact, the maximum number of

principal ports used by both systems is 2, i.e., the are 2INS's. This can be seen as a weak form of the above mentioned result by Khalil.

Sect. 5.5 addresses the issue of encoding the full π-calculus. The only features which are not covered by $\mathcal{C}$ are guarded choice and match prefixes; these are dealt with resp. in Sections 5.5.1 and 5.5.2. This part of the chapter is rather informal; the encoding becomes so technical that it would take way too long to expose it in full detail. Besides, doing so would not give much benefit, since guarded choice and match prefix are seldom considered essential.

It must be mentioned that Alexiev had already given an encoding of Fπ into a variant of multiport interaction nets; our encoding, quite different from his, has the advantage of using only "pure" multiport interaction nets, and of being able to cope with every feature of the π-calculus, in particular with replication.

Although we do not give any explicit proof, it is not hard to imagine how the concurrent behavior of multiport interaction nets can be faithfully encoded in the π-calculus; therefore, the main consequence of this chapter is that mINS's are just as expressive as the π-calculus. We would like to remark here that, contrarily to all other graphical formalisms used to represent the π-calculus, multiport interaction nets were not "built around" any process calculus, i.e., they are not *ad hoc* at all. Our encoding should be seen as a benchmark, a way to test the expressiveness of mINS's, rather than as yet-another-graphical-encoding of the π-calculus.

The Multiport Combinators

In this chapter we seek the multiport equivalent of the interaction combinators, i.e., a multiport interaction net system having the property of being *universal*. We have seen that, in order to define universality, we must first define the notion of *translation*.

Translations up-to. In the deterministic case, a translation from an INS $\mathcal{S}$ to an INS $\mathcal{S}'$ is a map Φ from the nets of $\mathcal{S}$ to the nets of $\mathcal{S}'$ verifying the following properties (below, μ is a generic net of $\mathcal{S}$):

(1) the active pairs of μ are in bijection with the active pairs of $\Phi(\mu)$;

(2) $\mu \to \nu$ implies $\Phi(\mu) \to^* \Phi(\nu)$.

In the non-deterministic case, (1) and (2) are insufficient to guarantee the correctness of a translation: nothing forbids $\Phi(\mu)$ from reducing to a net which has nothing to do with the translation of any reduct of μ.

For this reason, in Sect. 6.1 we introduce the notion of *translation up to*. If $\mathcal{S}$ and $\mathcal{S}'$ are two mINS's, and if $\sim$ is a congruence on the nets of $\mathcal{S}'$, we say that a map Φ from the nets of $\mathcal{S}$ to the nets of $\mathcal{S}'$ is a translation of $\mathcal{S}$ into $\mathcal{S}'$ up to $\sim$ iff, for any net μ of $\mathcal{S}$, the following holds:

(1) the active pairs of μ are in bijection with the active pairs of $\Phi(\mu)$;

(3) if $\Phi(\mu) \to \mu'$, then $\mu' \sim \Phi(\nu)$, where ν is the one-step reduct of μ obtained by reducing the active pair corresponding to the one reduced in $\Phi(\mu)$ (according to the bijection given by (1)).

Deterministic translations are translations up to $\simeq_\beta$ (or actually up to $\rightarrow^*$, if we allow precongruences). The stronger the congruence $\sim$, the more "convincing" a translation will be; the intention is that $\sim$ should be a suitable behavioral equivalence.

The meaning of property (3) is that, whenever $\mathcal{S}$ can be translated into $\mathcal{S}'$, $\mathcal{S}'$ is able to simulate the choices of $\mathcal{S}$ instantaneously: in fact, due to non-determinism, $\mu \rightarrow \nu$ in $\mathcal{S}$ may imply that μ has "made a choice", i.e., it has taken a non-confluent computational branch; property (1) guarantees that $\Phi(\mu)$ can "move" accordingly, and property (3) states that if μ' is the result of such "move", the choice has already been made, because μ' behaves exactly as $\Phi(\nu)$, even though it is not necessarily equal to it (as usual, it may take several steps of $\mathcal{S}'$ to syntactically simulate a step of $\mathcal{S}$).

The multiport combinators. In Sect. 6.2 we introduce the multiport combinators. These are a conservative extension of Lafont's interaction combinators; they form an infinite mINS (in the sense that the number of principal ports is unbounded), but for all $n \geq 1$, one can define a finite nINS with $n+2$ cells, the n-port combinators. The 1-port combinators are exactly the interaction combinators, and, for every $n \geq 1$, the n-ports combinators are strictly contained in the $n+1$-port combinators.

In Sect. 6.3 we define a notion of behavioral equivalence for the 2-port combinators, denoted by $\cong$. Just as the observational equivalence defined in the deterministic case, it is defined as the contextual-closure of a bisimilarity based on observable paths, although these need to be redefined in presence of several principal ports. In particular, it is not yet clear how the definition can be extended in case there are more than two principal ports; this is why we have preferred to restrict to the 2-port combinators. The main result of this section is Lemma 6.2: we deem *deterministic* certain reduction rules (in particular those of the 1-port combinators), and we prove that if $\mu \rightarrow^* \mu'$ by means of deterministic rules only, then $\mu \cong \mu'$.

Universality. Sect. 6.4 contains the proof of the universality of the 2-port combinators: any 2INS can be translated up to $\cong$ in the 2-port combinators. The translation we give is essentially a concurrent extension of Lafont's translation for the deterministic case; in fact, when one restricts to INS's, the translation essentially becomes Lafont's one [Laf97]. Lemma 6.2 is crucial: in fact, if $[\mu]$ is the translation of a net μ containing an active pair yielding the reduction $\mu \rightarrow \nu$, we have that $[\mu] \rightarrow \mu' \cong [\nu]$, because $\mu' \rightarrow^* [\nu]$ by means of deterministic steps only.

By the results of Sect. 5.4, Cπ can be encoded using a 2INS; therefore, in spite of their simplicity (4 cells and 10 rules), the 2-port combinators are at least as expressive as Cπ, which in turn is essentially as expressive as the full π-calculus.

Part I

Semantics of Interaction Nets

Chapter 1

Interaction Nets

1.1 Cells, wires, nets

Interaction nets are a graph-rewriting model of deterministic distributed computation, introduced by Lafont [Laf90]. They are inspired by the proof-nets of multiplicative linear logic [Gir87a, Gir87b, DR89, Laf95], and share many nice properties with them, with the advantage of being much more expressive.

1.1.1 The formal definitions

Formally, an interaction net is the union of two structures, a labelled, directed hypergraph, and an undirected graph. The directed edges will be called cells, and their label symbols; the undirected edges will be called wires. We now go through the details of the definition.

An *alphabet* Σ is a finite set of *symbols*, ranged over by α, β, each with an associated non-negative integer called its *arity*.

Let Σ be an alphabet such that the maximum arity of its symbols is m. We denote by Σ_n the subset of symbols of Σ of arity n. A *net* μ on Σ is a triple $(\mathsf{Ports}(\mu), \mathsf{Cells}(\mu), \mathsf{Wires}(\mu))$, where $\mathsf{Ports}(\mu)$ is a finite set, the elements of which are called the *ports* of μ, and

$$\begin{aligned} \mathsf{Cells}(\mu) &\subseteq \bigcup_{n=0}^{m} \Sigma_n \times \mathsf{Ports}(\mu)^{n+1} \\ \mathsf{Wires}(\mu) &\subseteq \mathcal{M}_2(\mathsf{Ports}(\mu)) \end{aligned}$$

where $\mathcal{M}_2(\mathsf{Ports}(\mu))$ is the set of multisets of $\mathsf{Ports}(\mu)$ of cardinality 2, and $\mathsf{Ports}(\mu)^k$ denotes the Cartesian product of k copies of $\mathsf{Ports}(\mu)$. The elements of $\mathsf{Cells}(\mu)$ and $\mathsf{Wires}(\mu)$ are called resp. the *cells* and *wires* of μ; they must satisfy the following constraints:

- in each cell, each port appears at most once;
- each port appears in exactly one wire, and can appear at most twice in $\mathsf{Cells}(\mu) \cup \mathsf{Wires}(\mu)$.

The ports of μ which appear only once in $\mathsf{Cells}(\mu) \cup \mathsf{Wires}(\mu)$ are called *free*. A free port can be principal, auxiliary, or neither, depending on the nature of the

other port to which it is connected by the only wire using it. The set of the free ports of a net is referred to as its *interface*. The set of all nets on an alphabet Σ is denoted by $\langle\Sigma\rangle$, and is ranged over by μ, ν.

Given a net μ, we can extract an undirected graph from it, which will be useful to formally speak of paths in a net (see Sections 1.2.4 and 2.1):

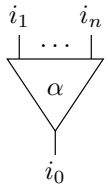
Definition 1.1 (Port graph) *The* port graph *of a net μ, denoted $\mathsf{PG}(\mu)$, is the undirected graph whose vertices are the elements of $\mathsf{Ports}(\mu)$, and such that, for $i, j \in \mathsf{Ports}(\mu)$, there is an edge between i and j iff one of the following (non-mutually exclusive) conditions holds:*

- $[i, j] \in \mathsf{Wires}(\mu)$*;*
- *i and j are resp. principal and auxiliary ports of the same cell.*

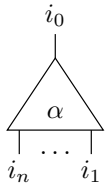
1.1.2 Graphical representations

The above definition of net has the advantage of being mathematically precise, but is far too heavy to work with; we now introduce the usual graphical representations of nets, which will be used in the rest of the thesis. In such representations, the ports of a net will be left implicit, and only cells and wires will be drawn.

A cell $(\alpha, i_0, i_1, \ldots, i_n)$ is represented as



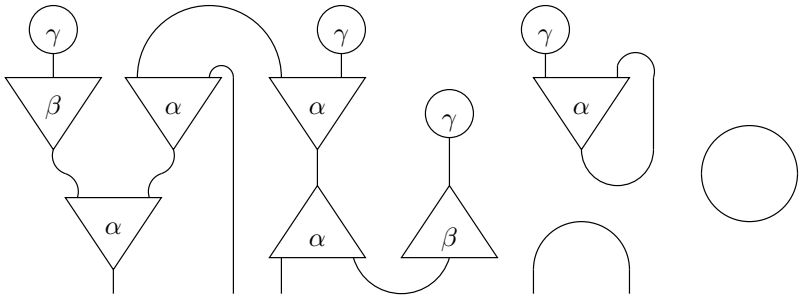
or



The names of the ports have been specified to make the reader understand which port is principal and which are auxiliary (and in which order); as we said, they will in general be omitted. In case $n = 0$, we use the following representation:

or the corresponding "upside down" version.

Wires will be simply pictured as... wires; the following is an example of graphical representation of a net on an alphabet with three symbols α, β, γ of resp. arity $2, 1, 0$:

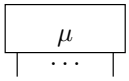


In graphical representations, the free ports of a net correspond to the "dangling" extremities of wires. The net above has for example 5 free ports, of which 1 is principal and 2 are auxiliary.

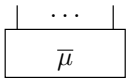
Observe that two nets have the same graphical representation iff they differ only by an injective renaming of the ports; we shall always consider nets modulo such renamings.

The fact that graphical representations "forget the names of ports" is a nice simplification, because most of the time these are completely irrelevant. The only ports one must pay attention to are the free ones: from these the formal version of a net can be uniquely recovered. Therefore, we shall consider equivalent any two graphical representations which, once the free ports are specified, yield the same formal net. In particular, wires can be yanked or twisted, and cells can be flipped or turned, as long as the free ports "stay where they are".

For example, when we have a fixed graphical representation of a net μ, the generic one being pictured as



it will sometimes be useful to consider the same representation rotated 180 degrees. We write this as

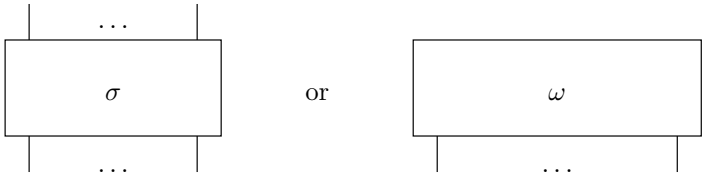


Of course, if μ has n free ports, numbered from 1 to n (we shall often consider this to be the case), and if we stipulate that the numbering increases "from left to right" in the first representation, then the numbering must increase "from right to left" in the second.

The notation $\overline{\mu}$ is actually rather sloppy, because it is not applied to μ itself, but to a particular graphical representation of it: strictly speaking, $\overline{\mu} = \mu$. However, this will not be a problem, as this notation will seldom be used, and whenever we make use of it, there will be no ambiguity as to its meaning. It has been introduced mainly for the purpose of Definition 1.2, and for other similar contexts where we need a handy way to denote 180-degree rotations of graphical representations of nets.

1.1.3 Wirings

A net containing no cells and no cyclic wires will be called a *wiring*. We shall represent the generic wiring as

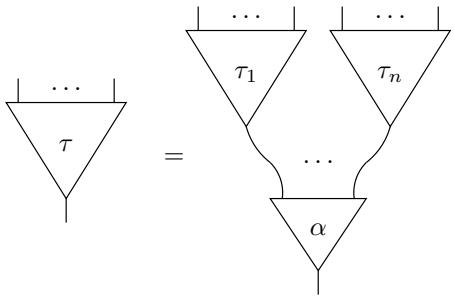


The following are examples of wirings:

Observe that a wiring has always an even number of free ports.

1.1.4 Trees

Trees are defined inductively as follows. A single zeroary cell is a tree with no leaves; a single wire is a tree with one leaf (it is arbitrary which of the two extremities is the root and which is the leaf); if $\tau_1, \ldots, \tau_n$ are trees with $k_1, \ldots, k_n$ leaves, and if α is cell of arity n, then the net

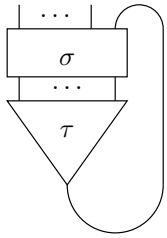


is a tree with $k_1 + \cdots + k_n$ leaves.

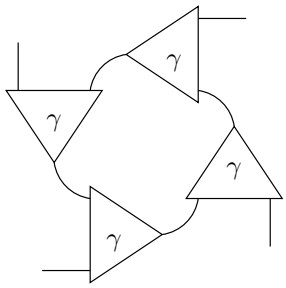
1.1.5 Vicious circles

A *vicious circle* is either a cyclic wire, or a net consisting of $n \geq 1$ cells $\alpha_1, \ldots, \alpha_n$, all of arity at least 1, such that, for all $1 \leq i < n$, the principal port of α_i is connected to an auxiliary port of α_{i+1}, and the principal port of α_n is connected to an auxiliary port of α_1.

With the notations introduced so far, a generic vicious circle can be represented as follows:



with the requirement that τ be "minimal", i.e., that no cell can be removed from τ without breaking the cycle. Here is an example of vicious circle, using only a cell γ of arity 2:

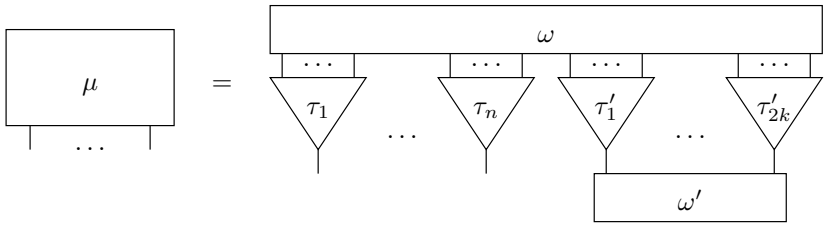


Notice that, because of its cyclic nature, a given vicious circle may not have a unique representation in terms of a tree and a wiring as above; in fact, a vicious circle with $n \geq 1$ cells may in general admit n different trees and wirings τ, σ representing it.

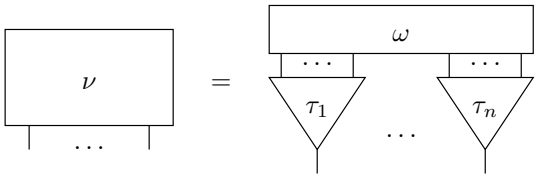
1.1.6 Active pairs and cut-free nets

A net consisting of two cells connected through their principal ports is called an *active pair*. If α and β are the two cells composing the active pair, we denote it by $\alpha \bowtie \beta$. A net containing no active pair and no vicious circle is said to be *cut-free*.[1]

It is not hard to verify that any net μ with n free ports and k active pairs and/or vicious circles can be decomposed (although not uniquely, by the above remark concerning vicious circles) in terms of trees and wirings as follows:



The wiring ω' accounts for the active pairs and vicious circles of the net; therefore, a cut-free net ν with n free ports admits the following decomposition, which this time is unique:

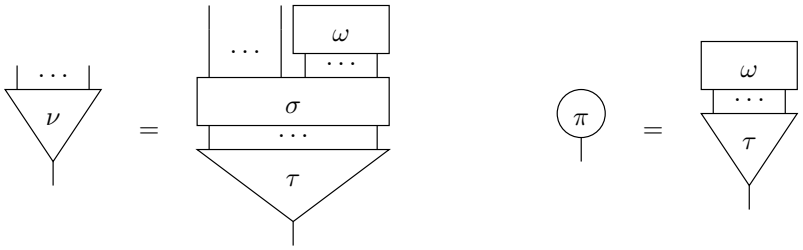


[1] *Reduced* in Lafont's terminology [Laf97].

1.1.7 Principal nets and packages

A principal net of arity n is either a single wire (in which case $n = 1$), or a cut-free net with n free auxiliary ports and 1 free principal port. If $n = 0$, we say that the net is a *package*. Principal nets can be seen as "compound" cells, and will be drawn just like ordinary cells. Notice that trees are particular examples of principal nets.

Following the decomposition of cut-free nets given above, a principal net ν and a package π take the following shape according to our notations:



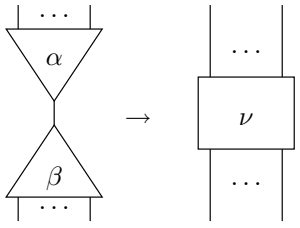
The set of all principal nets over an alphabet Σ is denoted by $\wp(\Sigma)$.

1.2 Interaction rules

The distinction between principal and auxiliary ports comes into play when defining the dynamics of nets. As a matter of fact, a net μ can be seen as one step of a (local) graph-rewriting process: according to some fixed *interaction rules*, a subnet of μ may be replaced by another subnet with the same interface, yielding a new net μ'. The process continues until no further rewriting is possible, i.e., until we reach a normal form. The key feature of interaction nets is that the left member of an interaction rule, i.e., the subnet which is susceptible of being replaced by another net, must be an active pair, and that for each active pair at most one interaction rule is defined.

1.2.1 Reduction and β-equivalence

Given an alphabet Σ and two cells $\alpha, \beta \in \Sigma$, an *interaction rule* for α and β associates to the active pair $\alpha \bowtie \beta$ a cut-free net $\nu \in \langle\Sigma\rangle$, together with a bijection between the free ports of ν and those of $\alpha \bowtie \beta$. We shall write interaction rules as follows:



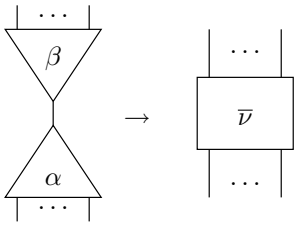
In this way, the bijection will never need to be made explicit, because it is clear from the graphical representation.

When we want to apply the above interaction rule to an active pair $\alpha \bowtie \beta$ contained in a net $\mu \in \langle\Sigma\rangle$, we first remove $\alpha \bowtie \beta$ from μ; this creates a number

of free ports, which are then connected to the free ports of ν, according to the bijection given by the rule. This operation can of course be described more precisely using the formal definitions given in Sect. 1.1.1, but we shall content ourselves with the above informal description, which we hope the reader will find clear enough.

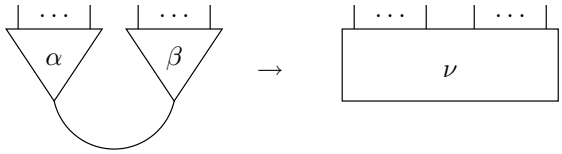
If μ' is the net resulting from μ after applying an interaction rule, we say that μ *reduces in one step* to μ', and we write $\mu \to \mu'$. We can then define the *reduction relation* $\to^*$ as the reflexive-transitive closure of $\to$. We say that two nets μ and μ' are *β-equivalent*, and we write $\mu \simeq_\beta \mu'$, iff there exists μ'' such that $\mu \to^* \mu''$ and $\mu' \to^* \mu''$.

Clearly, due to the essential lack of orientation of active pairs, the above rule can be rewritten as

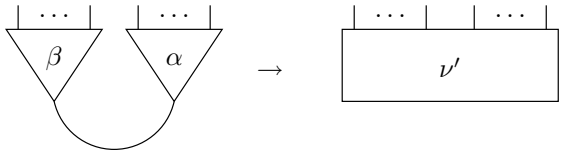


where we have used the notation introduced at p. 20.

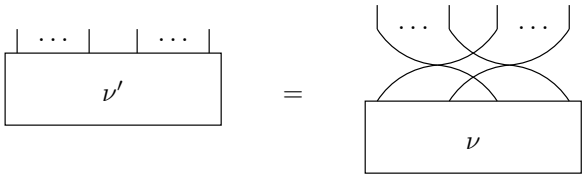
There is also the following alternative way of writing interaction rules, which will be used from time to time in the sequel:



Of course, in this case the rule can be rewritten as



where



Notice that interaction rules are purely local; if we add to this the fact that cells have exactly one principal port, and that there is at most one rule for each active pair, we immediately obtain

Proposition 1.1 (Strong confluence) *The relation $\to$ is confluent, i.e., if μ, μ_1, μ_2 are three nets such that $\mu \to \mu_1$ and $\mu \to \mu_2$, then there exists μ' such that $\mu_1 \to \mu'$ and $\mu_2 \to \mu'$.*

Proposition 1.1 means that the reduction process, i.e., the relation $\rightarrow^*$, is *strongly* confluent. Confluence implies that $\simeq_\beta$ is an equivalence relation, and that the system is deterministic in the sense that each net has at most one normal form. Strong confluence reinforces this determinism, because it implies that also the computation is unique, up to permutation of rules.

We remark here a substantial difference with respect to the λ-calculus, which is the absence of a meaningful concept of *strategy*, at least as far as the efficiency of reduction is concerned. All reductions leading from a net to its normal form have the same length and require the same work; in particular, if a net is normalizable, then it is strongly so.

Observe that vicious circles are deadlocked configurations with respect to reduction: they can never be removed from a net, since cells can only interact through their principal port. A net admitting a cut-free form (necessarily unique by confluence) is said to be *total*. Cut-free nets are the "true" normal forms of the reduction; they can be seen as the final result of a computation. On the other hand, non-total nets represent deadlocked or divergent computations.

Another difference with respect to the λ-calculus is attested by the following result:

Proposition 1.2 *A net μ is total iff all of its subnets are total.*

Proof. The "if" part is trivial: if μ is not total, then there obviously exists a subnet of μ which is not total, it is μ itself! For what concerns the "only if" part, assume that μ is total. By definition, and by confluence, there exists a unique cut-free net ν such that $\mu \rightarrow^* \nu$. Now, if a subnet μ_0 of μ were to be non-total, at least one of following would hold:

1. μ_0 contains vicious circles or irreducible active pairs, or reduces to a net containing them. But vicious circles and irreducible active pairs cannot be eliminated, so they would be present in ν, contrarily to our hypothesis;

2. μ_0 has an infinite reduction sequence; then, μ would also have an infinite reduction sequence, which is against our hypothesis because of strong confluence.

In any case we obtain a contradiction, thus μ_0 must be total. □

The above result tells us that being non-total in interaction nets is somewhat "stronger" than being non-normalizable in the λ-calculus: if a λ-term T contains a non-normalizable subterm, T may still be normalizable, an obvious example being $(\lambda y.x)\Omega$; on the contrary, if μ is a non-total net, any net containing it will still be non-total.

1.2.2 Interaction net systems

We can now give a formal definition of *interaction net system* (INS). To simplify matters, we assume that the free ports of nets are indexed with integers from 1 to n, so that whenever two nets have the same interface, there is a default bijection between their free ports (the identity on their indices).

Definition 1.2 (Interaction net system) *An* interaction net system *(INS for short) is a triple $(\Sigma, \mathcal{R}, \mathcal{N})$ where Σ is an alphabet, $\mathcal{R}$ a partial function from $\Sigma \times \Sigma$ to $\langle\Sigma\rangle$, and $\mathcal{N} \subseteq \langle\Sigma\rangle$ such that:*

- *for all $\alpha, \beta \in \Sigma$, if (α, β) is in the domain of $\mathcal{R}$, then $\mathcal{R}(\alpha, \beta)$ is a cut-free net with the same interface of $\alpha \bowtie \beta$, and $\mathcal{R}(\beta, \alpha)$ is also defined and equal to $\overline{\mathcal{R}(\alpha, \beta)}$;*
- *if $\mu \in \mathcal{N}$, and if $\mu \to \mu'$ according to the interaction rules given by $\mathcal{R}$, then $\mu' \in \mathcal{N}$.*

The set $\mathcal{N}$ of the above definition is called the set of *correct nets* of the system. Most of the time, the set of correct nets is omitted, i.e., an INS is defined simply as a couple $(\Sigma, \mathcal{R})$; in this case, it is assumed that $\mathcal{N} = \langle\Sigma\rangle$.

1.2.3 Typing and polarized systems

A typical way one can define the set of correct nets to be smaller than the totality of nets is through typing. Given a set of type constants, ranged over by T, we define *input types* and *output types*, denoted resp. by T^- and T^+. An alphabet Σ can be typed by assigning to each port of each cell a type constant plus an input/output specification. Then, we say that a net μ of $\langle\Sigma\rangle$ is *well-typed* iff each wire of μ connects ports of the same type but of opposite polarity.

Given a typed alphabet Σ, an INS $(\Sigma, \mathcal{R}, \mathcal{N})$ is said to be *typed* iff $\mathcal{N}$ is the set of well-typed nets on Σ, $\mathcal{R}$ is defined only on well-typed active pairs, and the interaction rules preserve the typing of the interface of the active pair they reduce.

A typed INS with only one type constant is said to be *polarized*. In this case, the type assignment is reduced to a simple input/output specification, and the requirement of a net to be well-typed can be reformulated by saying that the wires are directed, the type of a port being the incoming/outgoing nature of the connections it accepts.

1.2.4 Deadlock freeness

One of the nicest feature of interaction nets is the definability of deadlock-free (i.e., vicious-circle-free) fragments by means of purely structural conditions [Laf90], inspired to Danos&Regnier's correctness criterion for multiplicative linear logic [DR89]. As Lafont says [Laf98], this is indeed a beautiful example of transfer of concept from logic to computer science: logical correctness becomes deadlock freeness.

Given an alphabet Σ, we call *partitioning* of Σ a function P assigning to each cell of Σ a partition of its auxiliary ports. Remember our convention establishing that the auxiliary ports of a cell of arity n are numbered from 1 to n; if $\mathcal{P}(A)$ denotes the powerset of a set A, then P can be seen as a function from Σ to $\mathcal{P}(\mathcal{P}(\mathbb{N}))$, such that, if the arity of α is n, then $P(\alpha)$ is a partition of $\{1, \ldots, n\}$. A pair (Σ, P) where Σ is an alphabet and P a partitioning of Σ is called a *partitioned alphabet*. The following definition uses the port graph of a net (Definition 1.1):

Definition 1.3 (Switching) *Let (Σ, P) be a partitioned alphabet, and let $\mu \in \langle\Sigma\rangle$. A* switching *of μ is a subgraph of $\mathsf{PG}(\mu)$ such that, if α is a cell of μ, then for all $A \in P(\alpha)$ all edges connecting the ports of A to the principal port of α have been suppressed except one.*

Definition 1.4 (Deadlock-free net) *Let (Σ, P) be a partitioned alphabet. A net $\mu \in \langle\Sigma\rangle$ is* deadlock-free[2] *iff every switching of μ is acyclic.*

We call *deadlock-free* any INS $(\Sigma, \mathcal{R}, \mathcal{N})$ such that Σ is partitioned by some function P, $\mathcal{N}$ is the set of deadlock-free nets according to P, and the rules of $\mathcal{R}$ preserve deadlock-freeness. The terminology is justified by the following:

Theorem 1.3 (Lafont [Laf90]) *Let $(\Sigma, \mathcal{R}, \mathcal{N})$ be a deadlock-free INS, and let $\mu \in \mathcal{N}$. Then, for any μ' such that $\mu \to^* \mu'$, μ' does not contain vicious circles.*

Proof. A deadlock-free net does not contain vicious circles, for if it did, it is immediate to see that there would be a cyclic switching. But, by definition, deadlock freeness is preserved through reduction, hence the result. □

1.3 Examples

In this section we illustrate the expressive power of interaction nets by giving a few interesting examples of interaction net systems. These examples have been chosen so that they complement those already found in the existing literature. In Sect. 1.3.1 we treat to a certain depth multiplicative linear logic proof-nets, which are the source of the interaction nets paradigm, expanding to some extent part of a previous survey by Lafont [Laf95]. In Sect. 1.3.2 we introduce the interaction combinators and their symmetric version [Laf97], the latter being of fundamental importance for the rest of our work (see Chapter 3). As a proof of the expressive power of the symmetric combinators, we give what to our knowledge is the first explicit encoding of the **SK** combinators into the interaction combinators. We also show how to solve generic recursive equations in the interaction combinators (symmetric or not), using a fundamental construction due to Lafont, and briefly discuss the non-existence of fixpoint combinators. In Sect. 1.3.3 we use once again Lafont's construction to show how all partial recursive functions can be programmed into a simple interaction net system. This completes in some sense an example by Lafont showing how elementary arithmetical expressions involving addition and multiplication could be programmed, and is also a novelty with respect to the existing literature.

For other interesting examples, we refer the reader to the following pieces of work: Lafont's papers for Turing machines, one-dimensional cellular automata, and other more basic programming examples like list manipulation [Laf90, Laf97]; Ian Mackie and Jeorge Pinto's encoding of linear logic proof nets [MP02]; and Sylvain Lippi's encoding of the left-reduction λ-calculus [Lip02a].

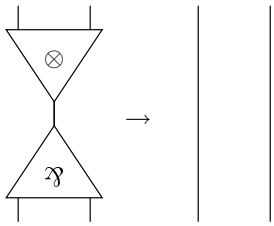
1.3.1 Multiplicative proof-nets

The prototypical example of interaction net system is given by the *pre-proof-structures* of multiplicative linear logic without units, or **MLL** [Gir87b, DR89].

[2] *Semi-simple* in Lafont's terminology. Lafont's original definition included also the notion of *simple* net, which would correspond to requiring all switchings to be acyclic *and connected*. However, this definition is notoriously problematic in the presence of zeroary cells: it is the old issue of defining proof-nets for multiplicative linear logic with units. A solution has recently been proposed by Dominic Hughes [Hug05], but it does not seem to be easily adaptable to the more general context of interaction nets. To avoid any problem, Lafont used an inductive definition [Laf90]. Anyway, since semi-simplicity already guarantees Theorem 1.3, we chose not to consider simplicity at all.

The alphabet $\Sigma_{\mathbf{MLL}}$ consists of the two binary cells

The function $\mathcal{R}_{\mathbf{MLL}}$ is defined on the active pair $\otimes \bowtie \parr$ as follows, and it is undefined on the other two active pairs:



We refer of the system $(\Sigma_{\mathbf{MLL}}, \mathcal{R}_{\mathbf{MLL}})$ as the "pre-proof-structures" of **MLL** because of the absence of typing. The typing system of multiplicative linear logic is not as simple as the typing systems introduced in Sect. 1.2.3: it does not consist simply of a set of dual type constants, and it involves a sort of polymorphism, in which the ports of a cell accept any type. Nevertheless, given a net $\mu \in \langle \Sigma_{\mathbf{MLL}} \rangle$, we may assign an **MLL** formula[3] to the ports of $\mathsf{Ports}(\mu)$ as follows:

- a wire connecting two auxiliary ports of μ, or an auxiliary port and a free port, or two free ports, is called an *axiom*; a wire connecting two principal ports of μ is called a *cut*; all other wires of μ are referred to as *simple*;
- if $i, j, k \in \mathsf{Ports}(\mu)$ are resp. the first and second auxiliary ports and the principal port of a $\otimes$ cell, and if the formulas assigned to i and j are resp. A and B, then the formula assigned to k must be $A \otimes B$;
- if $i, j, k \in \mathsf{Ports}(\mu)$ are resp. the first and second auxiliary ports and the principal port of a $\parr$ cell, and if the formulas assigned to i and j are resp. A and B, then the formula assigned to k must be $A \parr B$;
- if $i, j \in \mathsf{Ports}(\mu)$ are connected by an axiom or a cut, then the formulas assigned to them must be linear negations of each other;
- if $i, j \in \mathsf{Ports}(\mu)$ are connected by a simple wire, then the same formula must be assigned to them.

A net of $\langle \Sigma_{\mathbf{MLL}} \rangle$ that admits an assignment satisfying the above requirements is called a *proof-structure*; we denote the set of proof structures by PS. It is not hard to check that, since De Morgan duals are defined by $(A \otimes B)^\perp = B^\perp \parr A^\perp$ and $(A \parr B)^\perp = B^\perp \otimes A^\perp$, the set PS is stable under the interaction rule of $\mathcal{R}_{\mathbf{MLL}}$, so $(\Sigma_{\mathbf{MLL}}, \mathcal{R}_{\mathbf{MLL}}, \mathsf{PS})$ is an INS.

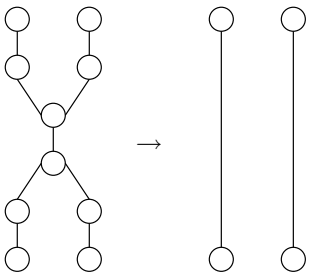
[3] We recall that the formulas of unit-free multiplicative linear logic (ranged over by A, B) are generated by the grammar $A, B ::= X \mid X^\perp \mid A \otimes B \mid A \parr B$, with $X, X^\perp$ ranging over a denumerable set of dual variables, and with linear negation, denoted $(\cdot)^\perp$, being involutive and defined by De Morgan's laws, knowing that the connectives $\otimes$ (tensor) and $\parr$ (par) are duals of each other.

As the reader may know, not all proof-structures are logically meaningful. By "logically meaningful" one usually means the following. We know that it is possible to define a function $(\cdot)^\bullet$ associating to each sequent calculus proof π of **MLL** a proof-structure $\pi^\bullet$. It is easy to see that this function is not injective; in fact, this lack of injectivity is the interesting property of the translation, because it comes from the the elimination of useless sequential information. However, $\pi^\bullet$ is not surjective either; the elements of PS which are in the image of $(\cdot)^\bullet$ are what one usually refers to as "correct", or "logically meaningful" proof-structures.

There is another, more interesting way of defining logical correctness though. By an easy argument (the number of cells strictly decreases after each reduction step) one can show that the reduction process always terminates already in the case of pre-proof-structures. Yet, it may terminate with a non-cut-free net, i.e., a net containing irreducible active pairs and/or vicious circles. The first are avoided when switching to proof-structures: the typing guarantees that irreducible active pairs never arise through reduction. On the contrary, the second are still possible, even though they are reduced to the cyclic wire only (all other vicious circles are not typable). The idea is then to say that "logically meaningful" means "satisfying cut-elimination", i.e., correct proof-structures should interact in a way that never produces deadlocks.

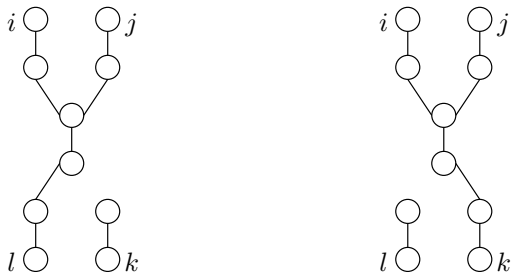
Girard's key idea was to isolate by purely geometrical means (with a so-called *correctness criterion*) a class of proof-structures, called *proof-nets*, satisfying cut-elimination; then, his sequentialization theorem showed that proof-nets are exactly the proof-structures in the image of $(\cdot)^\bullet$, so that the two definitions of logical soundness given above eventually coincide (provided we make the technical adjustment of adding the mix rule[4] to linear logic). In the context of interaction nets, we have seen that deadlock-freeness can be achieved by suitably partitioning the auxiliary ports of the alphabet $\Sigma_{\mathbf{MLL}}$.

In fact, knowing Danos and Regnier's version of the correctness criterion [DR89], we can choose our partitioning P by setting $P(\otimes) = \{\{1\},\{2\}\}$ and $P(\parr) = \{\{1,2\}\}$; then, the deadlock-free nets of Definition 1.4 are exactly the proof-nets of **MLL** (with the mix rule). The stability of deadlock freeness under reduction can be proved as follows. First of all, consider the port graphs of the two members of the interaction rule:



If we assume that the $\otimes$ cell is at the top of the active pair, then the left member of the rule has two switchings:

[4] In sequent calculus, the mix rule allows to prove $\Gamma, \Gamma' \vdash \Delta, \Delta'$ from $\Gamma \vdash \Delta$ and $\Gamma' \vdash \Delta'$.



Now, assume that the active pair is part of a deadlock-free net μ, and fix a switching G for it. We may assume that for our active pair G chooses the configuration which is on the left, the other one being perfectly symmetrical. Of course the "same" switching can be consider on the reduct μ': we simply ignore the $\parr$ cell which disappears, switching all other cells exactly as in G. We call G' such switching.

The first thing we remark is that i and j cannot be connected in G by a path other than the one present in the active pair, otherwise we would have a cycle, against our hypothesis of deadlock freeness. On the other hand, nothing forbids k and l to be connected in G, and this (lack of) connection is preserved in G', because the rule is local. If k and l are connected in G, then the active pair forms one connected component, and what is left of it keeps being a connected component in G'; if they are not, then there are two connected components, and again the situation is identical in G'. In both cases, the number of connected components of G and G' is the same.

We denote by v, e, c, and y resp. the number of vertices, edges, connected components, and cycles of G, and by v', e', c', and y' the same quantities in G'. Now, by hypothesis G has no cycle; therefore, by the Euler-Poincaré invariant[5], we have $c = v - e$. But $v = v_0 + 10$ and $e = e_0 + 8$, where v_0 and e_0 are the number of vertices and edges of G not accounted for by the active pair and the ports to which it is connected. We thus obtain $c = 2 + v_0 - e_0$. Since the rule is local, we have $v' = v_0 + 4$ and $e' = e_0 + 2$, so that using again Euler-Poincaré, we obtain $c' - y' = c$. But we have shown above that $c' = c$, so we must have $y' = 0$, which means that G' is acyclic. Since no particular assumption was made about the switching G', we have proved that μ' is deadlock-free.

This shows that the triple $(\Sigma_{\mathbf{MLL}}, \mathcal{R}_{\mathbf{MLL}}, \mathsf{PN})$ is an INS, where PN is the set of proof-nets, i.e., typable and deadlock-free nets of $\langle \Sigma_{\mathbf{MLL}} \rangle$. Denis Bechet [Bec98] has proved a very interesting property of the set PN, namely that it is a maximal deadlock-free subset of PS: if one adds to PN a proof-structure μ not satisfying the deadlock freeness condition (i.e., having a cyclic switching), then one can find a proof-net which, upon interaction with μ, produces a deadlock.

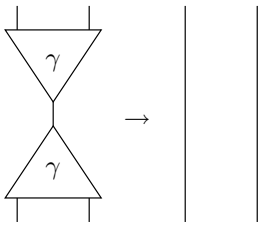
Another nice system can be defined by typing the cells of Σ with a single type T, this time in accord to what discussed in Sect. 1.2.3; as already said, in this case the system is *polarized*, and it is enough to consider directed wires. The typing is assigned as follows:

[5]We remind that the Euler-Poincaré invariant can be rephrased graph theoretically by saying that for any undirected graph with v vertices, e edges, c connected components, and y cycles, we have $v - e = c - y$.

The well-typed nets of $\langle \Sigma_{\mathbf{MLL}} \rangle$ according to this typing system are the pure (i.e., untyped) proof-structures of **MLLpol** (polarized multiplicative linear logic without units); if we consider the same partitioning given above, we obtain the pure polarized proof-nets of **MLLpol** (with mix). The stability of typing under reduction is obvious; the stability of deadlock freeness can be proved exactly as above.

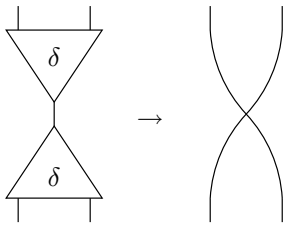
1.3.2 The interaction combinators

In the previous section we have seen how the pre-proof-structures of unit-free multiplicative linear logic can be seen as an interaction net system. In the definition, it is quite clear that the interaction between tensor and par does not "need" to know which is which; in other words, we could take a multiplicative proof-structure and "forget" the nature of cells, transforming all cells into a binary cell γ interacting with itself as follows:



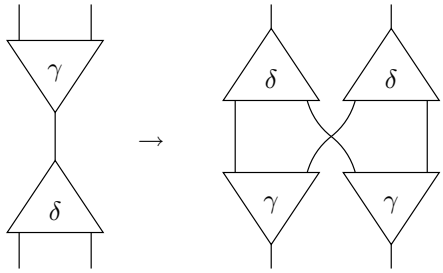
The dynamics of a net of $\langle \Sigma_{\mathbf{MLL}} \rangle$ obtained by turning every cell into a γ cell will be exactly that of a pre-proof-structure, with the advantage of no longer having irreducible cuts.

For semantical reasons (see Sect. 3.1.6), it is actually better to consider, instead of γ, a cell δ interacting as follows:

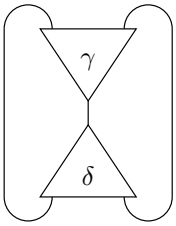


Of course the dynamics of such nets is nearly identical to that of nets composed of γ cells, and thus to multiplicative pre-proof-structures. In fact, the system $(\{\delta\}, \mathcal{R}_\delta)$ (where $\mathcal{R}_\delta$ is the rule above) corresponds to the "skeleton" of **MLL** cut-elimination: for the reader acquainted to categorical interpretations of linear logic, we can say that passing from **MLL** to this system is similar to passing from a $*$-autonomous to a compact closed category. The analogy is rather loose, because it forgets the units, but it is intuitively sound.

We can now think of making the two cells introduced so far interact with each other in a way which is reminiscent of the multiplication/comultiplication interaction of bialgebras:

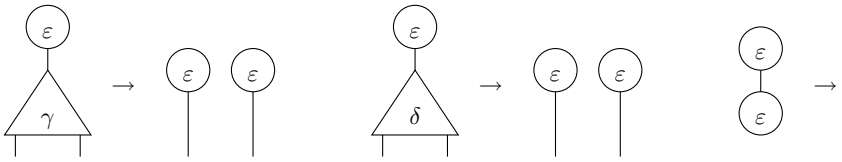


In spite of its simplicity, the system $(\{\gamma, \delta\}, \mathcal{R}_{\gamma,\delta})$ (where $\mathcal{R}_{\gamma,\delta}$ accounts for the three rules introduced do far) has an incredibly complex dynamics. In particular, in contrast to the previous two systems, it contains non-terminating nets like the following, which increases its size exponentially:



The one above is actually but a trivial example; as a matter of fact, the INS $(\{\gamma, \delta\}, \mathcal{R}_{\gamma,\delta})$ is Turing-complete.

We can add to it one more cell, an *eraser* cell, which interacts as follows:



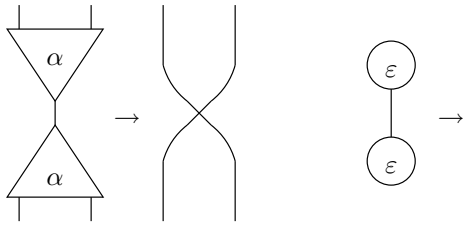
Notice that the rule $\varepsilon\varepsilon$ is forced by the requirement that right members of interaction rules be cut-free nets (the only cut-free net with an empty interface is the empty net).

If we set $\mathcal{R}$ to encompass all of the six rules introduced above, we can define the INS $\mathcal{C} = (\{\gamma, \delta, \varepsilon\}, \mathcal{R})$, which is called the system of the *interaction combinators* [Laf97]. We shall see that $\mathcal{C}$ is *universal*, in the sense that any other interaction net system can be translated into it (see Sect. 1.4).

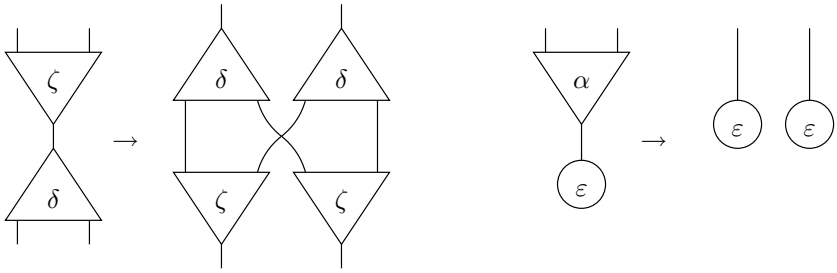
The interaction rules of $\mathcal{C}$ can be neatly divided into two groups: the *annihilations*, which tell how two cells carrying the same symbol interact, and the *commutations*, which tell how two cells carrying distinct symbols interact. Considering the universality of the interaction combinators, this has led Lafont to speculate that, with respect to interaction nets, the fundamental laws of computation are precisely annihilation and commutation, as opposed to substitution in the λ-calculus, or erasing and duplication in the **SK** combinators.

The subsystems built on the alphabets $\{\gamma, \varepsilon\}$ and $\{\delta, \varepsilon\}$ (which are called resp. the $\gamma\varepsilon$ and $\delta\varepsilon$ fragments of the interaction combinators) are also interesting. As discussed above, they are "isomorphic"; they can be seen as a sort of multiplicative pre-proof-structures plus *daimon*, i.e., the possibility of proving any formula. Moreover, in Sect. 1.4 we shall prove that any INS can be seen as an extension of the two fragments, in accord with the claim that interaction nets generalize the structure of the multiplicative rules of linear logic.

If we change the $\gamma\gamma$ interaction rule to be exactly as the $\delta\delta$ rule, we obtain another very interesting system, known as the *symmetric interaction combinators* (or simply *symmetric combinators*). Formally, the system is defined as $\mathcal{C}_{\text{sym}} = (\{\delta, \varepsilon, \zeta\}, \mathcal{R}_{\text{sym}})$, where $\mathcal{R}_{\text{sym}}$ describes the following rules (below, $\alpha \in \{\delta, \zeta\}$): the annihilations



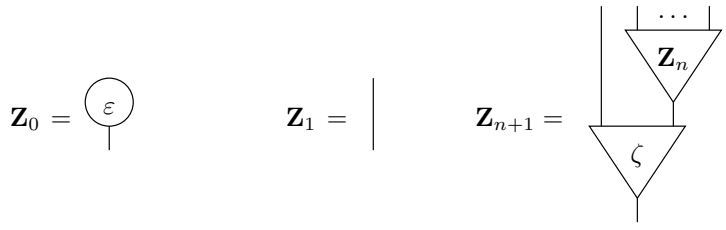
and the commutations



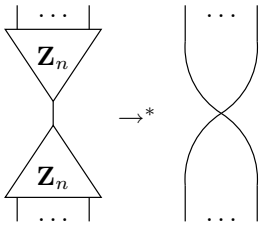
Even though the symmetric combinators lack the universality property of the interaction combinators, they are equally expressive, i.e., they are also Turing-complete, as we see shall see in a moment. Chapter 3 will be dedicated to an extensive semantical study of this system.

The Turing-completeness of the symmetric combinators is a corollary of Theorem 1.9; nevertheless, we shall explicitly prove it here, by providing an encoding of the (call by name) **SK** combinators in $\mathcal{C}_{\text{sym}}$.

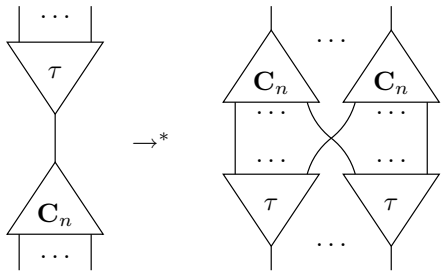
First of all, let us define a family of principal nets (actually trees) which we call *multiplexors*. For all $n \geq 1$, we pose



The reader can check that multiplexors have the following annihilation property, for all n (this can be proved by a straight-forward induction):

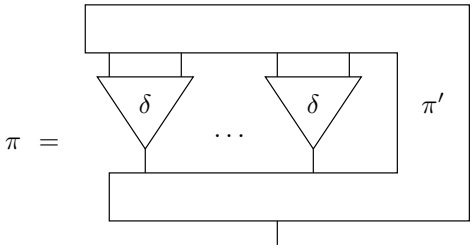


An identical construction can be done with δ cells, yielding a family of principal nets $\mathbf{C}_n$, called *copiers*, with the same annihilation property as multiplexors, and with the property that, if τ is a tree not containing δ cells, we have



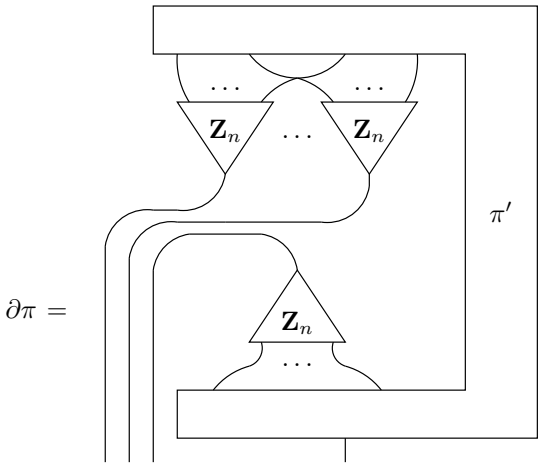
This is also easily established by induction on n and on the size of τ. Notice that, the rules for δ and ζ being identical, the same property holds for multiplexors whenever τ does not contain ζ cells.

We now introduce a fundamental construction due to Lafont [Laf97]. Take a generic package π containing n δ cells; we can always write π as

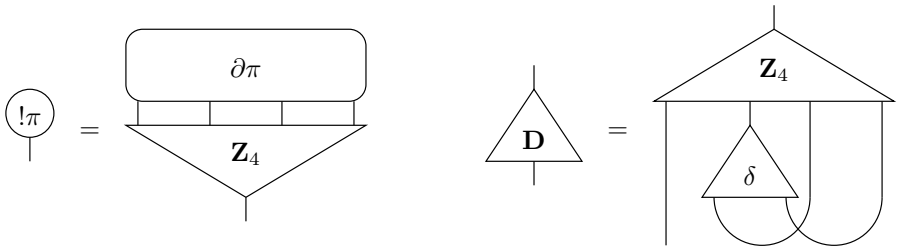


where π' contains no δ cell. We want to "abstract" the δ cells contained in π, forming a package $!\pi$ which does not contain δ cells, but from which π can be recovered.

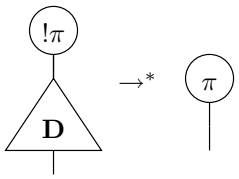
Using the multiplexors introduced above, we define



Then, we put



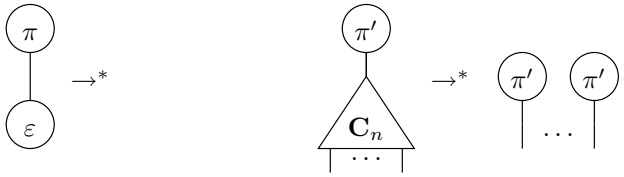
The reader can check that we have



The package !π is called the *code* of π, and the principal net **D** is called the *decoder*.

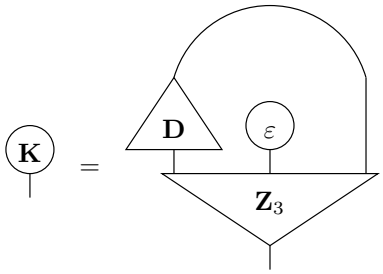
The following lemma is a straight-forward consequence of the rules concerning ε and of the properties of copiers stated above:

Lemma 1.4 (Erasure and duplication) *Let π be a package, and π' a package containing no δ cells. Then, we have*

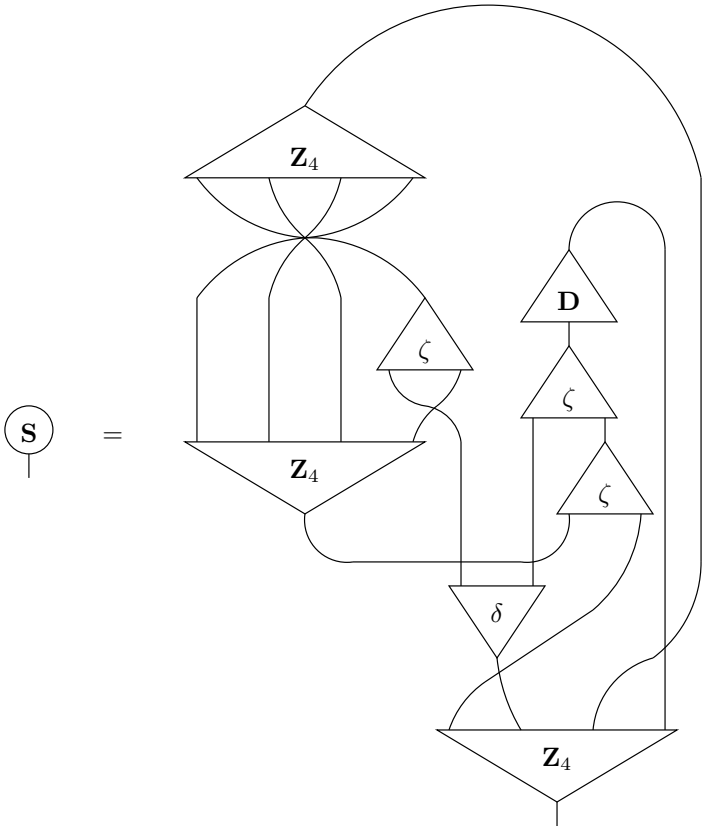


In particular, a package of the form !π can be erased and duplicated.

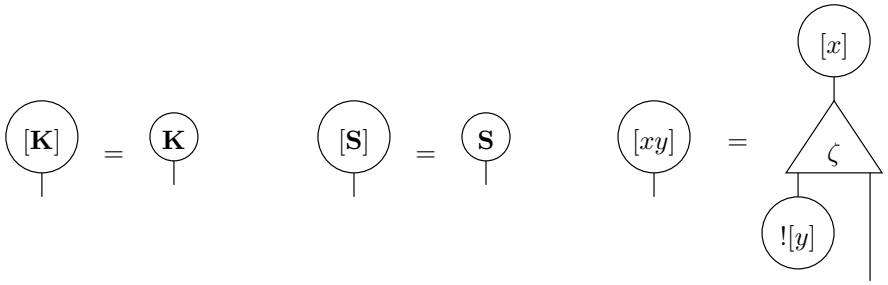
Now we build the two packages



and



and we define the translation $[\cdot]$ from **SK**-terms to nets:



If x, y are **SK**-terms, we write $x \succ y$ if x reduces to y through call-by-name reduction. For example, $\mathbf{K}(\mathbf{KSK})\mathbf{S} \succ \mathbf{KSK}$, but $\mathbf{K}(\mathbf{KSK})\mathbf{S} \not\succ \mathbf{KSS}$.

The translation defined above has the following property:

Proposition 1.5 *Let x, y be* **SK***-terms. Then, $x \succ^* y$ implies $[x] \to^* [y]$.*

Proof. It is enough to check that, for all terms x, y, z,

$$[\mathbf{K}xy] \to^* [x]$$

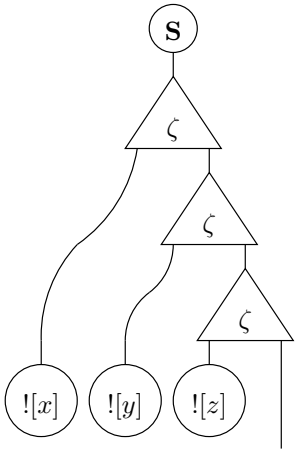
and

$$[\mathbf{S}xyz] \to^* [xz(yz)].$$

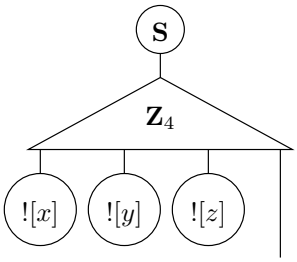
The first verification is easy and is left to the reader; Lemma 1.4 is needed for erasing. Here we shall concentrate on the second, which is more complex.

We have

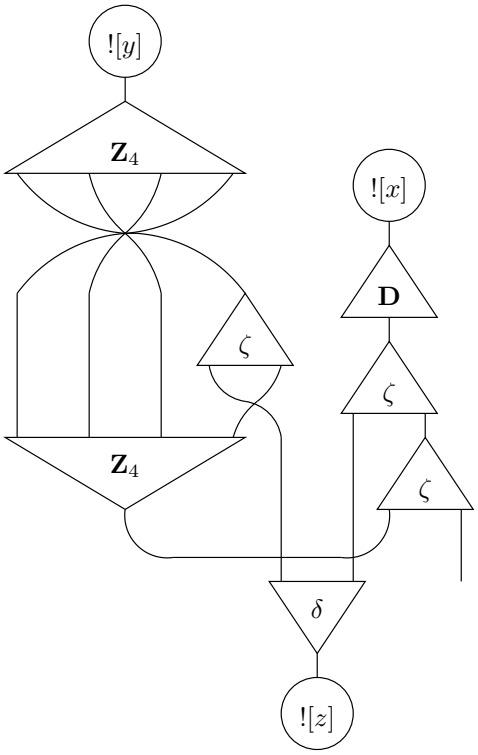
$[\mathbf{S}xyz] =$



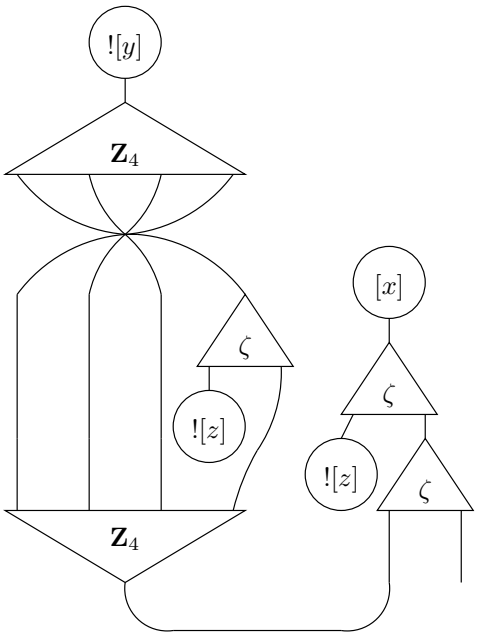
$=$



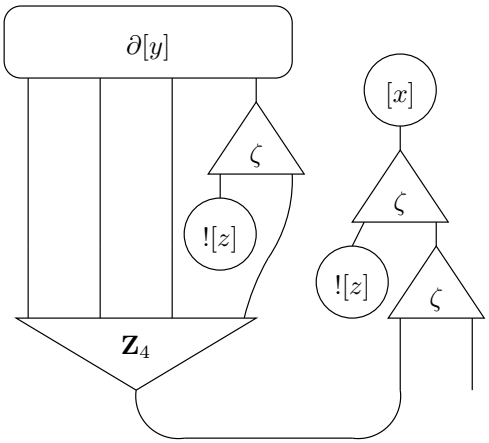
The $\mathbf{Z}_4$ tree annihilates with the $\mathbf{Z}_4$ tree contained in $\mathbf{S}$ (see p. 34), and we obtain



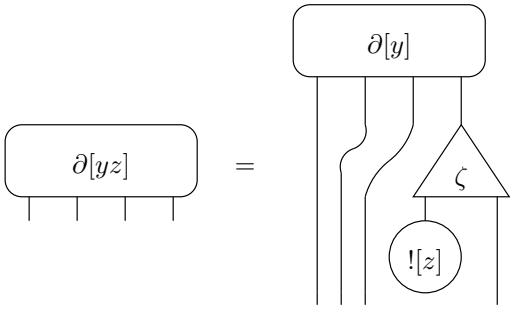
Now the decoder extracts $[x]$ from its code, and by Lemma 1.4 the code of $[z]$ is duplicated, so we get



At this point, the "topmost" $\mathbf{Z}_4$ tree annihilates with the $\mathbf{Z}_4$ tree inside $![y]$, and we are left with

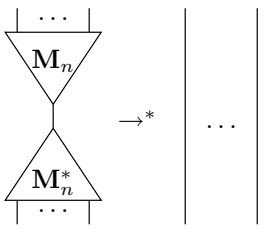


Remember that $![z]$ does not contain any δ cell; this means that



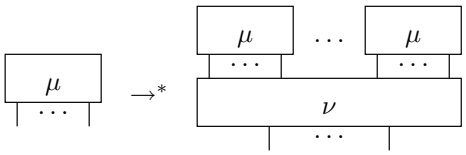
Therefore, the last net obtained is indeed equal to $[xz(yz)]$. □

Several remarks are in order here. First of all, multiplexors can be defined also in the interaction combinators using γ cells; because of the nature of the annihilation rule for these cells, there is actually the need for two dual families $\mathbf{M}_n$ and $\mathbf{M}_n^*$, annihilating as follows (see Sect. 6.4.2 for the details)



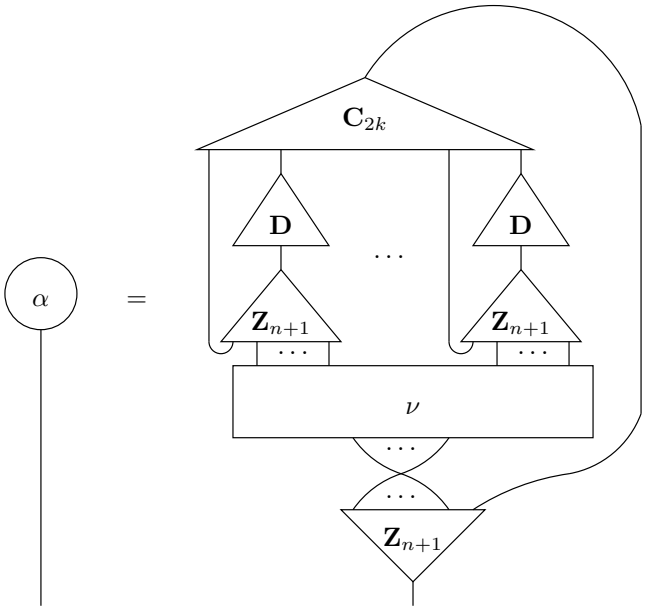
The fundamental annihilation/duplication properties of copiers stay the same (but notice that the symmetrical duplication property holding for $\mathbf{Z}_n$ nets fails for the $\mathbf{M}_n$ and $\mathbf{M}_n^*$ families). Thanks to this, Lafont !-construction can be defined on the interaction combinators as well, and our encoding of the **SK** combinators easily adapted to this system. Indeed, Lafont's originally introduced his !-construction on the interaction combinators, in order to show their universality (cf. Sect. 1.4); here we have presented it for the symmetric combinators simply because this latter system has a greater importance in our work than the interaction combinators.

We also remark that Lafont's !-construction is a very powerful tool for defining a sort of general recursion inside the interaction combinators (symmetric or not), analogous to the fix-point constructions in the λ-calculus. More precisely, thanks to it we can define a general procedure for building, given any net ν with sufficiently many free ports, a net μ satisfying the following property:

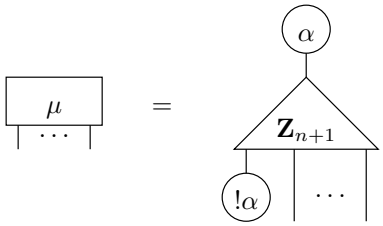


In other words, μ reduces to a net containing a certain (arbitrarily large) number of copies of itself.

In the case of the symmetric combinators (the case of the interaction combinators being perfectly analogous), we proceed as follows. Suppose that, in the above reduction, μ has n free ports, and the right member contains k copies of μ; then, given our net ν with $(k+1)n$ free ports, we construct a package α out of ν and the multiplexors, copiers, and decoders introduced above:



Then, the desired net μ can be built as

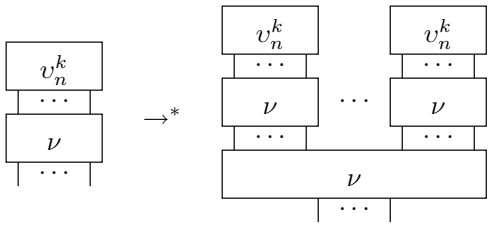


We leave it as an instructive exercise to the reader to verify that μ does indeed satisfy the recursive reduction above.

It is interesting to notice that, however, the above construction cannot be internalized; in other words, there is no fixpoint combinator in the interaction combinators. In the λ-calculus, the recursive equation we dealt with above would look something like

$$M \to^* N[M/x],$$

that is, M is a term reducing to a term containing it (we suppose that x is free in N). It is well known that a general solution to such recursive equations can be given in the λ-calculus by means of a fix-point combinator, i.e., a term $\mathbf{Y}$ verifying $\mathbf{Y}F \to^* F(\mathbf{Y}F)$ for any term F. In fact, it is enough to pose $M = \mathbf{Y}(\lambda x.N)$ to satisfy the above reduction. In the interaction combinators, this would correspond to solving the recursive equation by finding, for all $k, n \geq 1$, a family of nets υ_n^k with kn free ports such that, given any net ν with $(k+1)n$ free ports, we had



This is impossible because, even when $k = 1$, υ_n^1 should be able to duplicate ν regardless of its shape, and there is no way of building a universal duplicator in interaction nets (see Sylvain Lippi's Ph.D. thesis for a proof [Lip02b]). In the interaction combinators, only nets not containing δ can be duplicated; in the symmetric combinators, one can duplicate also those not containing ζ, but nothing can be done for nets containing both combinators. This is why Lafont had to devise his !-construction, which has an external nature, and cannot be internalized.

The ability of encoding general recursion is at the base of the universality of the interaction combinators, and of the polarized universality of the symmetric combinators (cf. Sect. 1.4). In any case, it shows that the expressive power of both systems is potentially very high, and indeed we know it to be Turing-complete, as the encoding of the **SK** combinators shows. Notice that, not surprisingly, in that encoding Lafont's !-construction plays a crucial rôle.

1.3.3 Recursive functions

As another example of the power and versatility of interaction nets, we use the constructions introduced in the previous section to show an INS in which all recursive functions can be programmed. If f is a partial function from $\mathbb{N}^k$ to $\mathbb{N}$, we denote by $\mathrm{dom}f$ the domain of f, and we write $f(x)\uparrow$ if $x \notin \mathrm{dom}f$. A function $f : \mathbb{N}^k \to \mathbb{N}$ is total just if $\mathrm{dom}f = \mathbb{N}^k$.

We remember that, if Nat is the set of all partial functions from $\mathbb{N}^k$ to $\mathbb{N}$ for all k, the set Rec of the recursive functions is the smallest subset of Nat such that:

Projections: for all $k \geq 1$ and $1 \leq i \leq k$, the total function $\pi_i^k : \mathbb{N}^k \to \mathbb{N}$ such that $\pi_i^k(x_1, \ldots, x_k) = x_i$ is in Rec;

Addition: the total function $\mathsf{add} : \mathbb{N}^2 \to \mathbb{N}$ mapping $(x, y) \in \mathbb{N}^2$ to $x + y$ is in Rec;

Multiplication: the total function $\mathsf{mul} : \mathbb{N}^2 \to \mathbb{N}$ mapping $(x, y) \in \mathbb{N}^2$ to $x \times y$ is in Rec;

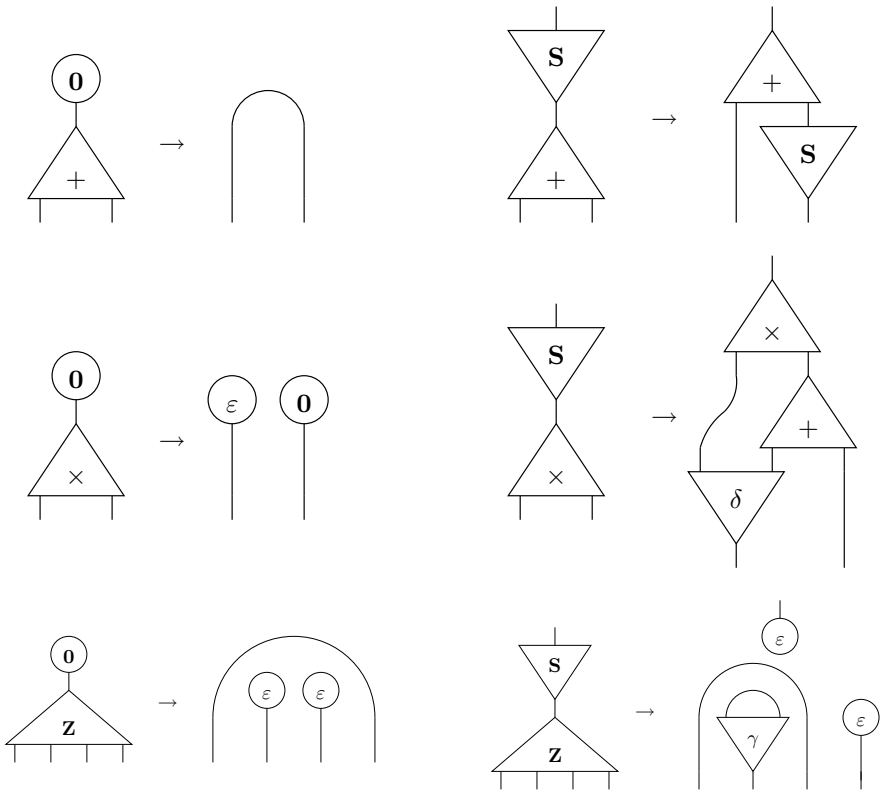
Composition: if $h : \mathbb{N}^n \to \mathbb{N}$ and $g_1 : \mathbb{N}^k \to \mathbb{N}, \ldots, g_n : \mathbb{N}^k \to \mathbb{N}$ are in Rec, then the function $f : \mathbb{N}^k \to \mathbb{N}$ defined as follows is in Rec:

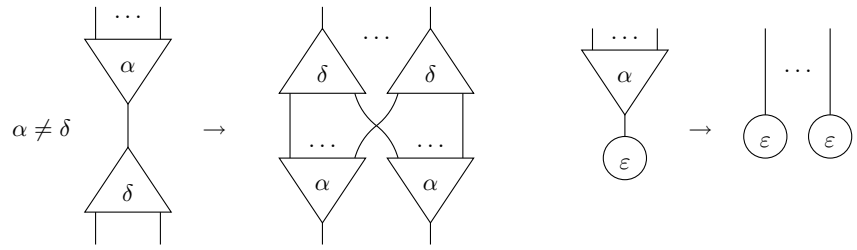
$$\forall \vec{x} \in \mathbb{N}^k,\ f(\vec{x}) = \begin{cases} h(g_1(\vec{x}), \ldots, g_n(\vec{x})) & \text{if, } \forall i,\ \vec{x} \in \mathrm{dom} g_i, \\ & \text{and } (g_1(\vec{x}), \ldots, g_n(\vec{x})) \in \mathrm{dom} h \\ f(\vec{x})\uparrow & \text{otherwise} \end{cases}$$

Unbounded search: if $g : \mathbb{N}^{k+1} \to \mathbb{N}$ is in Rec, then the function $f : \mathbb{N}^k \to \mathbb{N}$ defined as follows is in Rec:

$$\forall \vec{x} \in \mathbb{N}^k,\ f(\vec{x}) = \begin{cases} y & \text{if } g(\vec{x}, y) = 0, \\ & \text{and } \forall z \leq y,\ (\vec{x}, z) \in \mathrm{dom} g \\ f(\vec{x})\uparrow & \text{otherwise} \end{cases}$$

We now define the alphabet $\Sigma = \{\gamma, \delta, \varepsilon, \mathbf{0}, \mathbf{S}, +, \times, \mathbf{Z}\}$, where γ, δ, and ε are the interaction combinators (Sect. 1.3.2), and the arities of $\mathbf{0}$, $\mathbf{S}$, $+$, $\times$, and $\mathbf{Z}$ are resp. 0, 1, 2, 2, and 4. The rules $\mathcal{R}$ defined on Σ are the following:





The γ, δ, and ε cells interact between them exactly as in the interaction combinators; all other active pairs are left undefined. We call $\mathcal{S}_{\text{rec}}$ the INS $(\Sigma, \mathcal{R})$.

The multiplexors $\mathbf{M}_n/\mathbf{M}_n^*$, the copiers $\mathbf{C}_n$, and the decoder $\mathbf{D}$ can obviously be defined in $\mathcal{S}_{\text{rec}}$; Lafont's !-construction also applies to this system, with its fundamental property that a package of the form $!\pi$ can be erased and duplicated.

The intuitions behind the cells and rules of $\mathcal{S}_{\text{rec}}$ are the following:

- the interaction combinators are needed for structural manipulations (multiplexing/demultiplexing, erasing, copying, etc.);
- the $\mathbf{0}$ and $\mathbf{S}$ cells are the unary integers constructors, representing resp. 0 and the successor; the cells $+$ and $\times$ represent addition and multiplication: the interaction rules concerning them are nothing but a graphical form of the fundamental equations recursively defining these two operations:

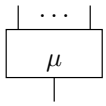
$$\begin{array}{rcl} x+0 & = & x \\ x+\mathbf{S}(y) & = & \mathbf{S}(x+y) \end{array} \qquad\qquad \begin{array}{rcl} x\times 0 & = & 0 \\ x\times \mathbf{S}(y) & = & x\times y+x \end{array}$$

  Notice that erasing and duplication are needed for multiplication because in the first equation x is not used, while in the second equation it is used twice;
- the cell $\mathbf{Z}$ represents a sort of "test for zero" construct: it waits for an integer x on its principal port, and carries two other values y_0 and y_1, which occupy two of its auxiliary ports; the other two auxiliary ports are the result, and an extra "control port". If x is zero, the cell returns y_0, erases y_1, and sends an eraser to the control port; if x is not zero, the cell returns y_1, erases y_0, and sends a γ package to the control port.

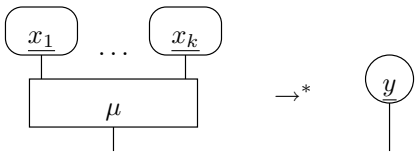
By the way, because of the universality of the interaction combinators, we know that $\mathbf{0}$, $\mathbf{S}$, $+$, $\times$, and $\mathbf{Z}$ could actually be encoded using only γ, δ, and ε; these extra cells can therefore be seen as "syntactic sugar", added to make the task of encoding recursive functions easier and more readable.

First of all, we give our interpretation of natural numbers. For all $x \in \mathbb{N}$, we define a package $\underline{x}$, called *numeral*, as follows:

A net $\mu \in \langle \Sigma \rangle$ is said to be *functional of arity* k iff it has a distinguished free port, called the *output*, and k free ports, called *inputs*; graphically, if μ is such a net, we shall represent inputs at the top, and the output at the bottom, as follows:



We shall say that a functional net μ of arity k *represents* a recursive function $f : \mathbb{N}^k \to \mathbb{N}$ iff, for all $\vec{x} = (x_1, \ldots, x_k) \in \mathbb{N}^k$, $\vec{x} \in \mathrm{dom} f$ and $f(\vec{x}) = y$ implies



whereas $f(\vec{x})\!\uparrow$ implies that the net on the left hand side is not total. A recursive function $f : \mathbb{N}^k \to \mathbb{N}$ is *representable* in $\mathcal{S}_{\mathrm{rec}}$ iff there exists a functional net of arity k representing f.

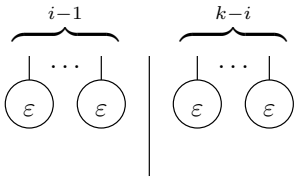
Proposition 1.6 *Every recursive function is representable in* $\mathcal{S}_{\mathrm{rec}}$.

We shall not go into the details of proving Proposition 1.6; we shall just give the idea of the proof, which we hope the reader will find to be convincing enough. Proposition 1.2 will play a fundamental rôle; the following result will also be often used in the sequel:

Lemma 1.7 *Numerals can be erased by* ε *cells and duplicated by* δ *cells.*

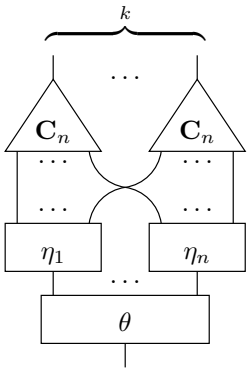
Proof. A straight-forward induction. □

We start by programming the base functions of Rec. By Lemma 1.7, the projection π_i^k can be represented by a net of the form

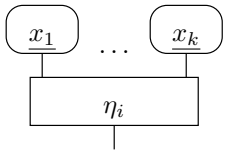


We invite the reader to check that addition and multiplication are represented by the following two nets:

Composition is encoded in the natural way: if $f : \mathbb{N}^k \to \mathbb{N}$ is defined by composing $h : \mathbb{N}^n \to \mathbb{N}$ with n functions $g_i : \mathbb{N}^k \to \mathbb{N}$, and if θ and η_i (for $1 \leq i \leq n$) represent resp. h and g_i, then the net



represents f. This is a consequence of Proposition 1.2: given $\vec{x} \in \mathbb{N}^k$, if one of the g_i is not defined on $\vec{x}$, then the net

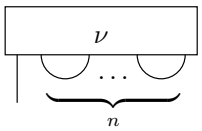


is not total, which means that the whole net representing f applied to $\vec{x}$ is non-total (Lemma 1.7 is used for copying the arguments).

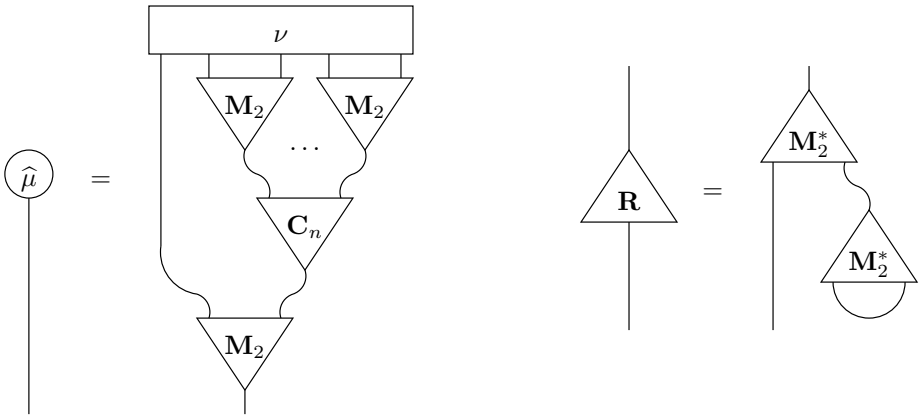
Unbounded search is encoded using the general recursion construction presented at the end of Sect. 1.3.2. In fact, defining $f : \mathbb{N}^k \to \mathbb{N}$ by unbounded search on $g : \mathbb{N}^{k+1} \to \mathbb{N}$ is equivalent to using the following recursive definition, in which a helper function h is defined:

$$\begin{aligned} h(\vec{x}, y) &= \texttt{if } g(\vec{x}, y) = 0 \texttt{ then } y \texttt{ else } h(\vec{x}, y+1) \\ f(\vec{x}) &= h(\vec{x}, 0) \end{aligned}$$

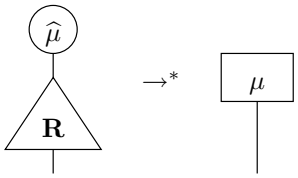
Given the functional net η representing g, the representation of h can be constructed as follows. First of all, we need to introduce a variant of the !-construction which, given a net μ with one free port, yields a package (i.e., a cut-free net) $\widehat{\mu}$, from which μ can be recovered. We know that in general, if such a net μ has n active pairs/vicious circles, we can write it as



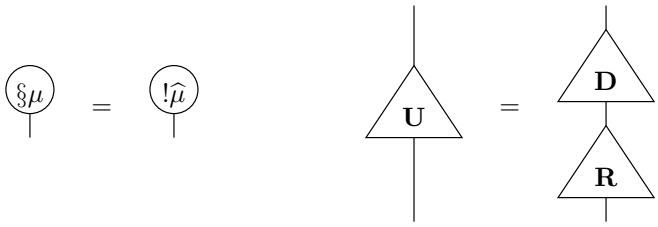
where ν is cut-free. So we pose



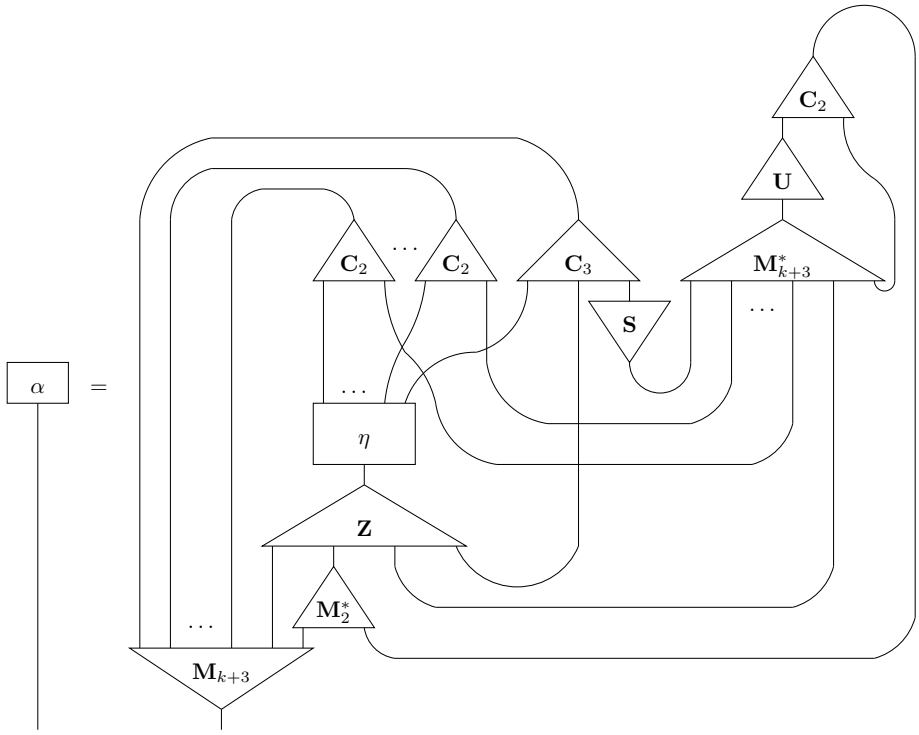
from which it is evident that $\widehat{\mu}$ is a package, and that



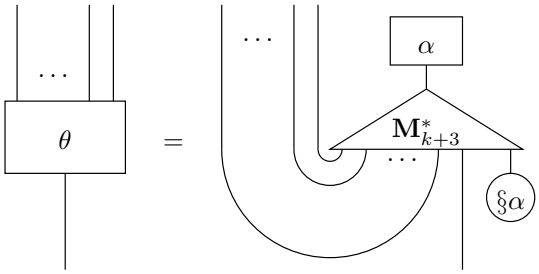
In the sequel, we shall use the following abbreviations:



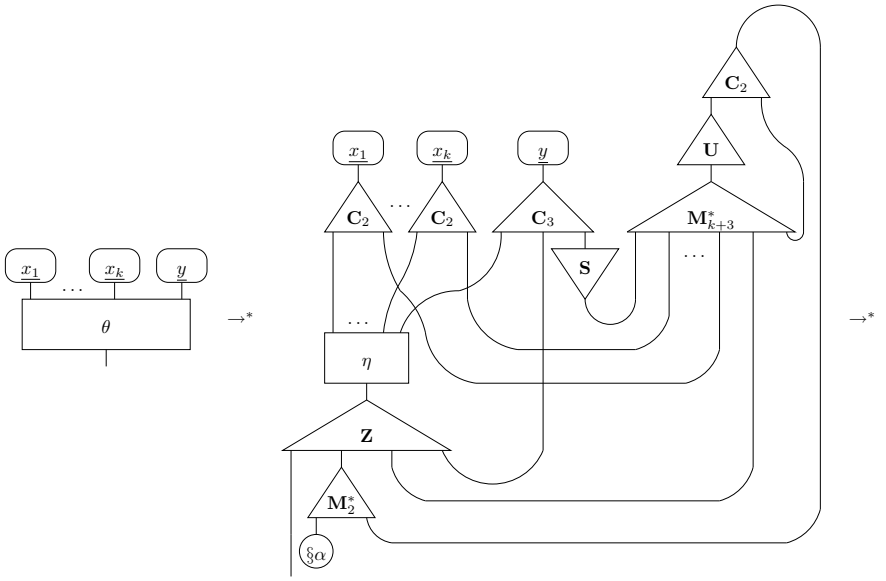
Now we can define

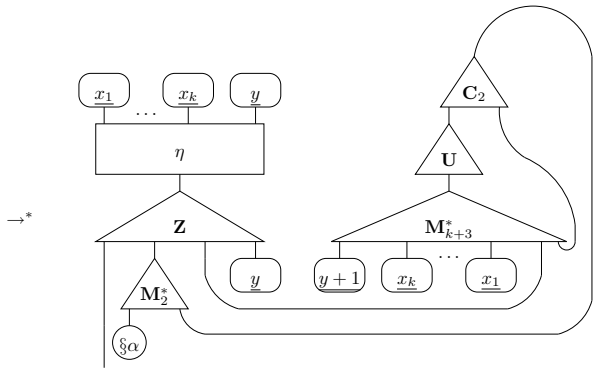


Then, we pose

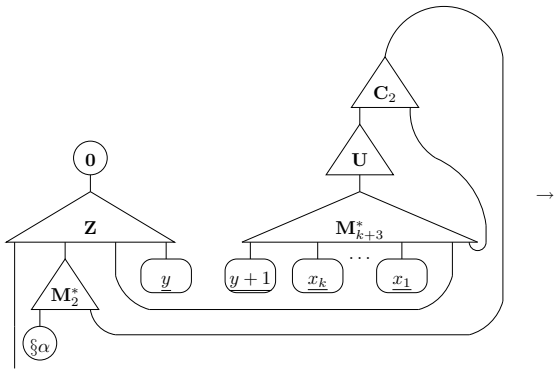


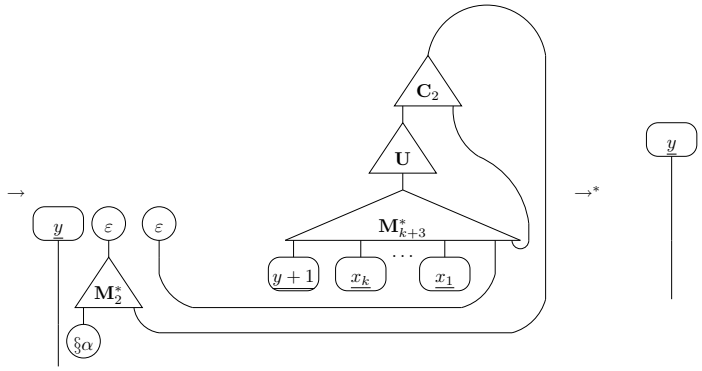
We claim that θ represents the helper function h defined above. This can be proved by checking a few reductions steps:



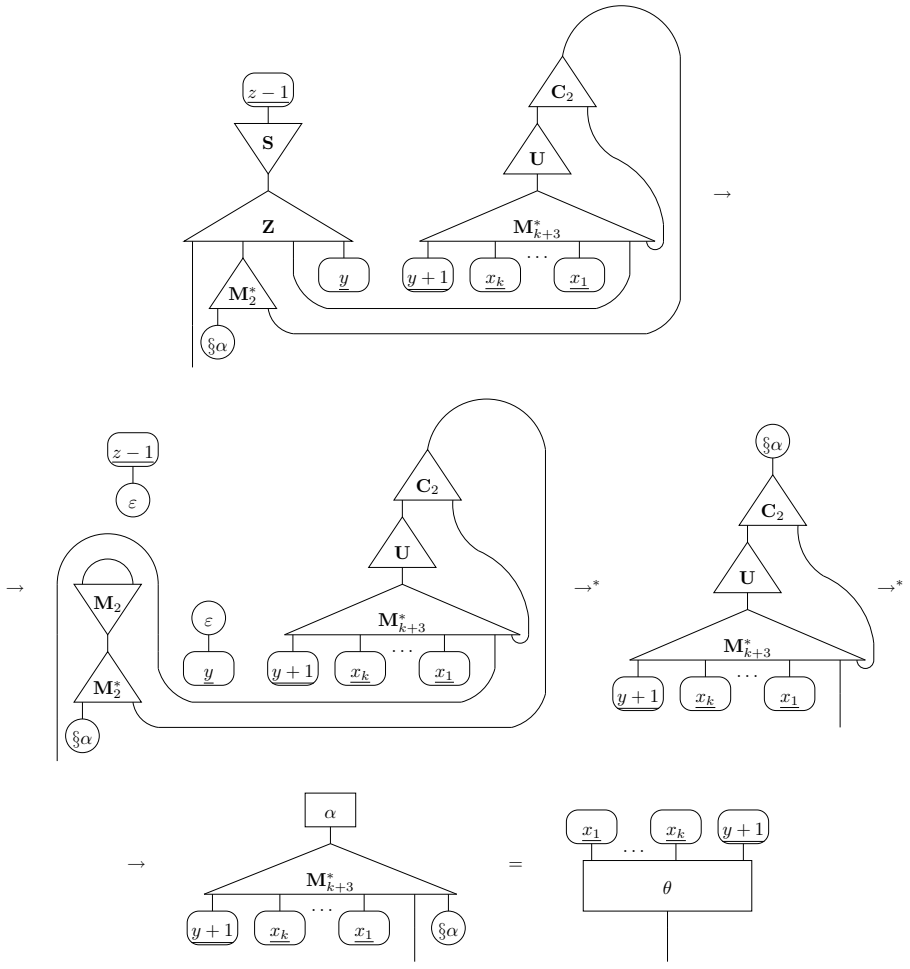


At this point, we can concentrate on the net η applied to the numerals $\underline{x_1}, \ldots, \underline{x_k}, \underline{y}$, since the rest of the above net is cut-free. If $g(\vec{x}, y)\uparrow$ then this net is non-total, and by Proposition 1.2 the whole net is non-total, which is correct, because in this case h is undefined. If $(\vec{x}, y) \in \mathrm{dom} g$, then there are two possibilities. The first is that $g(\vec{x}, y) = 0$; then, by hypothesis the above net reduces to





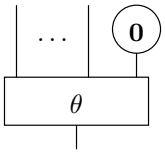
which is exactly what we want, since in that case $h(\vec{x}, y) = y$. The second possibility is that $g(\vec{x}, y) = z \neq 0$, in which case we do a recursive call on $h(\vec{x}, y+1)$. In fact, this time we have



Of course, if there is some $z > y$ such that $g(\vec{x}, z)\uparrow$, the above net will be non-total; the same will happen if there is no $z \geq y$ such that $g(\vec{x}, z) = 0$, because this

would yield an infinite reduction. In both situations, this is perfectly correct, because in neither case h is defined.

Now, after having verified that θ represents h, it should be clear that the net



represents our function f defined by unbounded search.

1.4 Interaction combinators and universality

1.4.1 Translations

We have introduced the interaction combinators (and their symmetrical variant) in Sect. 1.3.2, and we have made repeated allusions to their "universality". In order to state precisely what it means for an interaction net system to be universal, we first need the notion of *translation*. The main idea is that the existence of a translation from an INS $\mathcal{S}$ to an INS $\mathcal{S}'$ should not simply mean that $\mathcal{S}'$ can "encode" $\mathcal{S}$, or in other words that $\mathcal{S}'$ is at least as expressive as $\mathcal{S}$; it rather ought to mean that the fundamental computational properties of $\mathcal{S}$ can also be simulated by $\mathcal{S}'$, in particular its complexity and degree of parallelism. Then, we shall say that an INS is *universal* just in case any other INS can be translated into it.

In the following, if Σ, Σ' are two alphabets and Φ is a function from Σ to $\wp(\Sigma')$, we say that Φ is *arity-preserving* iff Φ maps cells of arity n to principal nets of arity n. Notice that such a function can obviously be extended into a function from $\langle\Sigma\rangle$ to $\langle\Sigma'\rangle$, still denoted Φ: given a net $\mu \in \langle\Sigma\rangle$, simply define $\Phi(\mu)$ to be the net in which each cell α in μ is replaced by $\Phi(\alpha)$, keeping the connections between ports exactly as they are in μ.

Definition 1.5 (Translation, [Laf97]) *Let $\mathcal{S} = (\Sigma, \mathcal{R})$ and $\mathcal{S}' = (\Sigma', \mathcal{R}')$ be two INS's, and let Φ be an arity-preserving function from Σ to $\wp(\Sigma')$. We say that Φ is a* translation *from $\mathcal{S}$ to $\mathcal{S}'$ iff, for all $\alpha, \beta \in \Sigma$ such that $\mathcal{R}(\alpha, \beta)$ is defined, we have $\Phi(\alpha \bowtie \beta) \to^* \Phi(\mathcal{R}(\alpha, \beta))$.*

The above definition is formulated without taking correct nets into account, or rather, the INS's are supposed to be such that the set of correct nets coincides with all possible nets. Obviously this is not restrictive at all: when we say that an INS $\mathcal{S}$ can be translated into an INS $\mathcal{S}'$, we assume that the correct nets of $\mathcal{S}$ can be translated (in the sense of Definition 1.5) into correct nets of $\mathcal{S}'$.

If Φ is a translation from $(\Sigma, \mathcal{R})$ to another INS such that, for all $\alpha \in \Sigma$, $\Phi(\alpha)$ contains at least one cell, then Φ is said to be *strict*. In the following, we shall assume that all translations are strict.

Observe that, if Φ is a translation from $\mathcal{S}$ to $\mathcal{S}'$, and if μ is a net of $\mathcal{S}$, then $\Phi(\mu)$ is cut-free iff μ is. In particular, $\Phi(\mu)$ contains the same number of active pairs as μ; by the simulation property stated in Definition 1.5, if $\mu \to^* \mu'$, then $\Phi(\mu) \to^* \Phi(\mu')$, so not only $\mathcal{S}'$ encodes $\mathcal{S}$ through Φ, but it also simulates its

degree of parallelism. Moreover, each rule of $\mathcal{S}$ is simulated by $\mathcal{S}'$ in a constant number of steps; if M and K are resp. the minimum and maximum number of steps taken by $\mathcal{S}'$ to simulate an interaction rule of $\mathcal{S}$, and if a reduction $\mu \to^* \mu'$ in $\mathcal{S}$ takes n steps, then the corresponding reduction $\Phi(\mu) \to^* \Phi(\mu')$ takes m steps in $\mathcal{S}'$, where $Mn \leq m \leq Kn$. Therefore, the time complexity of reduction is preserved by the translation; a similar fact holds for space complexity, in terms of number of cells. Notice that all of these properties may fail (for stupid reasons) in case Φ is not strict.

1.4.2 Universality

The following result is the object of [Laf97]:

Theorem 1.8 (Universality) *Any INS can be translated into the interaction combinators.*

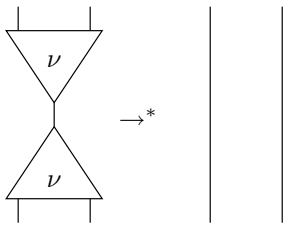
We shall not enter into the details of the proof of Theorem 1.8; in fact, in Sect. 6.4 we shall give a detailed proof of the universality of the multicombinators, of which the above result is a particular case. As expected, Lafont's !-construction (Sect. 1.3.2) plays a crucial rôle in the proof, because the reduct of a generic active pair $\alpha \bowtie \beta$ may contain occurrences of α and β themselves, generating a recursive reduction of the kind described at the end of Sect. 1.3.2.

The symmetric combinators, which will be the object of Chapter 3, do not enjoy Theorem 1.8. Instead, they verify a property that we refer to as *polarized universality*:

Theorem 1.9 (Polarized universality) *Any polarized INS can be translated into the symmetric combinators.*

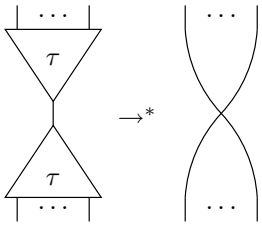
What the symmetric combinators really lack is the ability to simulate a cell interacting with itself, i.e., a rule reducing an active pair of the form $\alpha \bowtie \alpha$. The typical problematic example is the $\gamma\gamma$ rule of the interaction combinators:

Proposition 1.10 *In the symmetric combinators, there is no principal net ν of arity 2 such that*



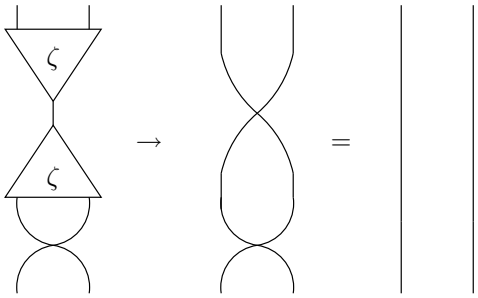
The above result is a consequence of the following lemma, which can be proved by means of a straight-forward induction:

Lemma 1.11 *Let τ be a tree of symmetric combinators. Then, we have*



Now, remember that a principal net ν of arity 2 is a tree plus a wiring connecting some of the leaves except two, which are left free. When ν interacts with itself, because of Lemma 1.11 the pairs of leaves connected by the wiring in one copy of ν exactly match those in the other copy, resulting in cyclic wires; the only two free leaves match each other, yielding a configuration which, if we neglect the cyclic wires, is identical to the right member of the $\delta\delta$ (and $\zeta\zeta$) rule, and thus different from that of the $\gamma\gamma$ rule.

If an INS is polarized, we only care about polarized interactions, which cannot be defined between a cell and itself. This asymmetry is exploited in the proof of Theorem 1.9. For instance, the $\otimes/\parr$ rule of Sect. 1.3.1 (or rather its polarized version), which is structurally identical to the $\gamma\gamma$ rule, can be trivially simulated in the symmetric combinators, for example using two ζ cells, one of which has its auxiliary ports "twisted":



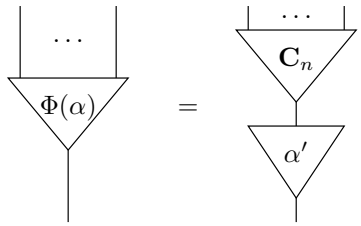
It is possible to see that virtually any "traditional" computational model can be encoded into a polarized INS. Important examples include Turing machines, one-dimensional cellular automata, **SK** combinators, the λ-calculus, and multiplicative exponential linear logic proof-nets. For example, following Ian Mackie and Jorge Sousa Pinto [MP02], we know that these latter can be encoded into the interaction combinators; the same encoding works for the symmetric combinators, up to minor adjustments which basically amount to "twisting wires" as in the example above. In Sect. 1.3.2 we gave a direct encoding of the call-by-name **SK** combinators into the symmetric combinators, which is another way of showing that, even though failing to satisfy full universality, this system has the same expressive power as the interaction combinators.

1.4.3 The $\delta\varepsilon$ fragment

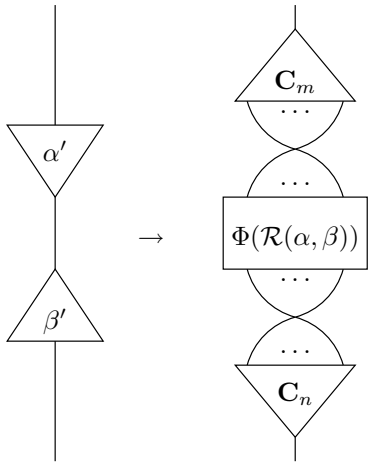
In this section we shall use the notion of translation introduced above to show that any INS can be seen as an extension of the $\delta\varepsilon$ fragment of the (symmetric) interaction combinators.

Proposition 1.12 *Let $\mathcal{S} = (\Sigma, \mathcal{R})$ be an INS, such that $\delta, \varepsilon \notin \Sigma$ (if δ or ε belong to Σ, simply rename them), and let Σ' be the alphabet composed of δ, ε, of resp. arity 2 and 0, and of α' of arity 1 for all $\alpha \in \Sigma$. Then, $\mathcal{S}$ can be translated into an INS $\mathcal{S}'$ whose alphabet is Σ'.*

PROOF. We use the multiplexors $\mathbf{C}_n$ introduced in Sect. 1.3.2 (actually we called them *copiers* in that context, but what is interesting to us here is their annihilation property, which is the same as that of the $\mathbf{Z}_n$ family). Given a cell $\alpha \in \Sigma$ of arity n, we define $\Phi(\alpha)$ as follows:

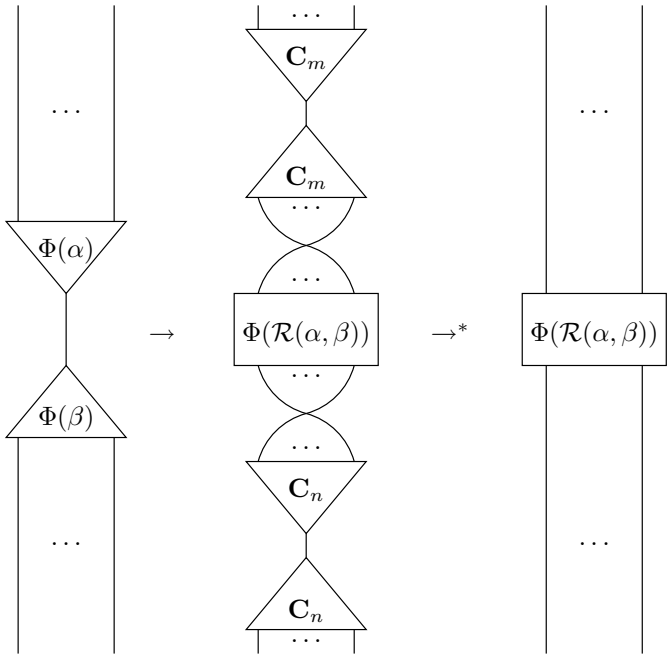


Then, we define the interaction rules of $\mathcal{S}'$, whose alphabet is Σ', according to the rules of $\mathcal{S}$; if α, β are two cells of Σ of resp. arity m and n, we pose:



δ and ε cells interact between them exactly as in the $\delta\varepsilon$ fragment, while their interaction with the other cells of Σ' is left undefined.

Following these definitions, if α, β are two cells of Σ of resp. arity m and n, by the annihilation property of $\mathbf{C}_n$ nets, we have



which shows that Φ is a translation from $\mathcal{S}$ to $\mathcal{S}'$. □

Of course the above proof can be adapted without problems to the $\gamma\varepsilon$ fragment: we just need to use $\mathbf{M}_n$ and $\mathbf{M}_n^*$ nets instead of $\mathbf{C}_n$ nets, and we do not need to cross the wires in the definition of the reduction rules.

Proposition 1.12 also shows that the arity of the cells of an alphabet is practically irrelevant: any INS can be translated into an INS in which all cells but two have arity 1; in particular, there is no need for cells of arity greater than 2. The interest of using cells with higher arities is that rules usually become simpler, i.e., the right members contain less cells, as the translation of Proposition 1.12 shows.

The $\delta\varepsilon$ (and $\gamma\varepsilon$) fragment can be seen as a sort of "pure arity" system: its dynamics simply consists in multiplexing/demultiplexing wires. We observe that this is nothing but the dynamics of **MLL** proof-structures. A generic INS simply adds to this the possibility of defining "arbitrary connectives", i.e., symbols (which determine the behavior) plus an arity (which is given by the multiplexors, i.e., **MLL** basic connectives).

Chapter 2

Observational Equivalence

2.1 Path-based observational equivalence

In the following, we fix a generic untyped INS $(\Sigma, \mathcal{R})$, and we take the terms "cell", "net", and "rule" to implicitly mean resp. "cell of Σ", "net of $\langle\Sigma\rangle$", and "rule of $\mathcal{R}$".

2.1.1 Straight paths and interaction paths

Our observational equivalence will be based on the fundamental notion of *straight path*, inspired by the corresponding notion in linear logic proof-nets [DR95]. The formal definition uses the *port graph* of a net, introduced in Definition 1.1.

Definition 2.1 (Straight path) *Let μ be a net. A path ϕ (not necessarily simple) of $\mathsf{PG}(\mu)$ is* straight *iff:*

(non-bouncing) *if ϕ contains a sequence of the form iji, then i and j are ports of the same cell, and they are connected by a wire in μ (i.e., there is a vicious circle);*

(non-twisting) *if i, j, k are resp. the principal port and two different auxiliary ports of the same cell, then ϕ does not contain the sequence jik.*

Seen directly on cells, the requirement that a path is non-bouncing means that we forbid paths like those of Figures 2.1a and 2.1b. Similarly, the non-twisting requirement excludes the paths like that shown in Fig. 2.1c. In other words, when a straight path enters a cell through one of its auxiliary ports, it exits through its principal port, and when it enters a cell through its principal port, it exits through one of its auxiliary ports.

By the way, the graph $\mathsf{PG}(\mu)$ is needed for formal purposes only; in the sequel, we shall freely speak of a "straight path of μ" meaning "straight path of $\mathsf{PG}(\mu)$", using $\mathsf{PG}(\mu)$ only in formal definitions. The same will be done for the other kinds of paths we shall introduce.

Definition 2.2 (Bouncing cell) *A cell α is* 0-bouncing *iff there exists a cell β such that in $\mathcal{R}(\alpha, \beta)$ there is a straight path between two free ports (not necessarily distinct) corresponding to the auxiliary ports of β, i.e., we have*

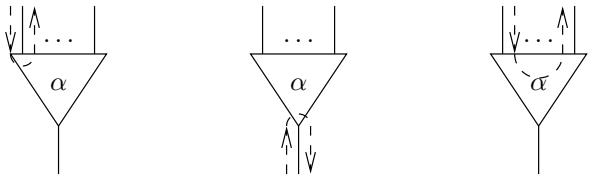
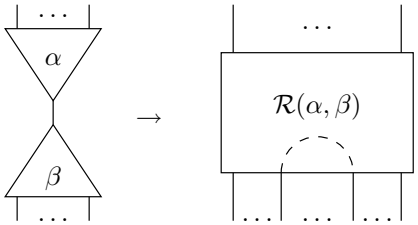
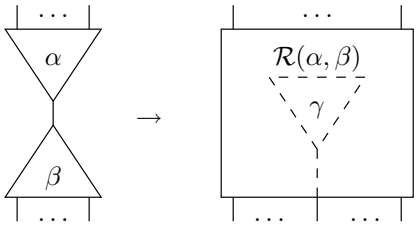


Figure 2.1: a,b. Bouncing paths. c. Twisting path.



For $n > 0$, we say that a cell α is n-bouncing *iff there exists a cell β such that in $\mathcal{R}(\alpha,\beta)$ there is a tree whose root is one of the free ports corresponding to the auxiliary ports of β which contains an $(n-1)$-bouncing cell. In other words, we have*



where γ is an $(n-1)$-bouncing cell. A cell is bouncing iff it is n-bouncing for some n.

The cell **0** in the system $\mathcal{S}_{\text{rec}}$ introduced in Sect. 1.3.3 is an example of bouncing cell (it is in fact 0-bouncing, because of the rule with the + cell). On the other hand, in the interaction combinators (symmetric or not) there are no bouncing cells; by universality, bouncing cells are thus not necessary as far as the expressive power of an interaction net system is concerned.

Definition 2.3 (Interaction path) *Let μ be a net. A path ϕ (not necessarily simple) of $\mathsf{PG}(\mu)$ is called an* interaction path *iff:*

(weakly non-bouncing) *if ϕ contains a sequence of the form iji, then either i and j are ports of the same cell, and they are connected by a wire in μ (i.e., there is a vicious circle), or j is the principal port of a bouncing cell and i is* not *an auxiliary port of the same cell;*

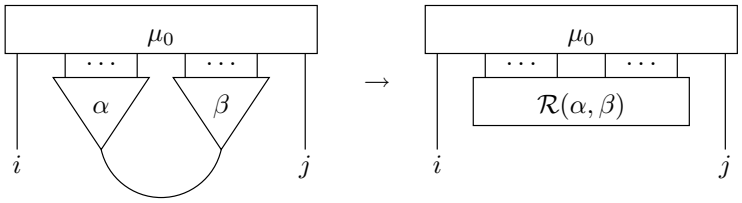
(non-twisting) *if i, j, k are resp. the principal port and two different auxiliary ports of the same cell, then ϕ does not contain the sequence jik.*

Seen directly on nets, interaction paths are straight paths allowing configurations like that of Fig. 2.1b in the case of a bouncing cell. Because of the absence of bouncing cells, in the interaction combinators (symmetric or not) straight paths and interaction paths coincide. The same happens in multiplicative exponential linear logic proof-nets seen as interaction nets, i.e., sharing graphs; this is why all paths mentioned in the Geometry of Interaction are always non-bouncing.

The main property of interaction paths is that of being preserved under anti-reduction:

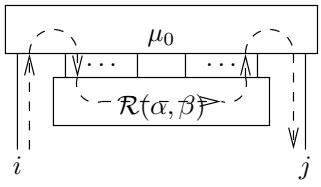
Proposition 2.1 *Let $\mu \to^* \mu'$, and let i, j be two free ports (not necessarily distinct) of μ and μ' (reduction preserves the interface, so the ports of μ' can be unambiguously identified with those of μ). If i and j are connected by an interaction path in μ', then they are connected by an interaction path in μ as well.*

Proof. It is obviously enough to prove the result in case $\mu \to \mu'$. The reduction can then be supposed to have the following form:

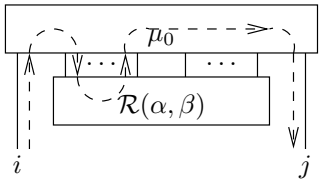


where, for brevity, only the free ports i and j are shown. There are three mutually exclusive situations:

- The interaction path in μ' does not go through $\mathcal{R}(\alpha, \beta)$. This case is trivial.
- The interaction path in μ' goes through $\mathcal{R}(\alpha, \beta)$, and it is of the following shape:

  This case too is trivial: an interaction path obviously exists in μ as well, it simply goes through the active pair.
- The interaction path in μ' goes through $\mathcal{R}(\alpha, \beta)$, and it is of the following shape:

This means that in $\mathcal{R}(\alpha, \beta)$ there is either a straight path or a bouncing cell "on the side" of the auxiliary ports of α. But then β is bouncing, so an interaction path exists in μ as well. The case in which the path uses only the interface of $\mathcal{R}(\alpha, \beta)$ corresponding to the auxiliary ports of β is identical; similarly, if the path uses both interfaces, "wiggling" a number of times between $\mathcal{R}(\alpha, \beta)$ and μ_0, then both cells are bouncing, and the result holds as well.

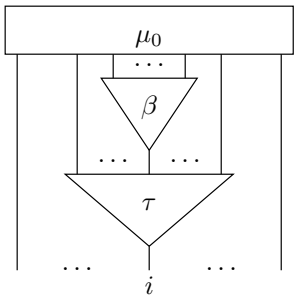
□

2.1.2 Observable paths

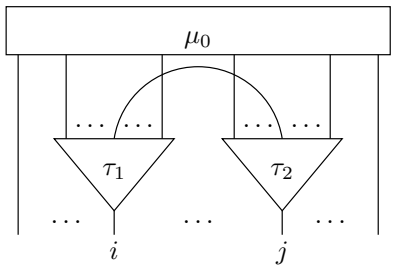
Interaction paths are preserved also under reduction whenever they cross no active pair. Upon these kind of paths is based our notion of observability.

Definition 2.4 (Observable path) *Let μ be a net, and i a free port of μ. We say that there is an* observable path *starting from i iff there exists a free port j of μ (not necessarily distinct from i) such that i and j are connected by a non-empty interaction path crossing no active pair. In this case, we say that the free port i is* immediately observable*, and we write $\mu{\downarrow_i}$.*

It is perhaps useful to "visualize" what it means for a free port to be immediately observable. Indeed, it is a straight-forward consequence of the definition that, if $\mu{\downarrow_i}$, only two (non-mutually exclusive) situations are possible: either μ is of the form



where β is a bouncing cell, or μ is of the form



where, if $i = j$, then $\tau_1 = \tau_2$ and the wire shown actually connects two leaves of the same tree. Of course, if $i \neq j$, then $\mu{\downarrow_i}$ implies $\mu{\downarrow_j}$. Notice also that one of τ_1 or τ_2 (or both!) may be equal to a wire, so an observable port need not be principal.

We remark that, since they do not contain active pairs, and since interaction rules are completely local, observable paths are preserved under reduction, as anticipated above:

Proposition 2.2 *Let μ be a net such that $\mu\downarrow_i$, and let $\mu \to^* \mu'$. Then, $\mu'\downarrow_i$.*

Proof. Obvious. □

Definition 2.5 (Observability predicates) *Let μ be a net, and i a free port of μ. We say that i is* observable*, and we write $\mu\Downarrow_i$, iff $\mu \to^* \mu'$ and $\mu'\downarrow_i$. If a port i is not observable, i.e., no reduct μ' of μ is such that $\mu'\downarrow_i$, then we say that i is* blind*, and we write $\mu\Uparrow_i$.*

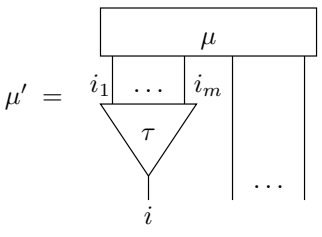
If we have $\mu\Downarrow_i$ for some free port i of μ, then we say that μ is observable, and we write $\mu\Downarrow$. On the other hand, if for all free ports i of μ we have $\mu\Uparrow_i$, then we say that μ is blind, and we write $\mu\Uparrow$.

Using observable paths, one can build a "proto-semantics" for interaction nets: to a net μ with n free ports, numbered from 1 to n, we associate a matrix with boolean coefficients $[\mu]$ such that $[\mu]_{ij} = 1$ iff μ reduces to a net containing an observable path from the free port j to the free port i. Of course, if $\mu \to^* \mu'$, then $[\mu] = [\mu']$. This interpretation, even forgetting its naïveness, does not yield a denotational semantics because it does not give rise to a congruence: if C is a context (see the following section), then $[\mu] = [\mu']$ does not imply $[C[\mu]] = [C[\mu']]$. It is however the prototype for the Geometry of Interaction semantics (see Sect. 3.3): there, the coefficients of matrices belong to a more complex structure, which is able to take into account very precisely the reduction of nets and their interaction with contexts. This only works nicely for the interaction combinators (symmetric or not), which is nevertheless quite satisfying thanks to their universality.

While observability can be altered by putting a net into a context, blindness sort of "propagates" inside a net, and blind free ports are "dull" with respect to interaction, as shown by the results below. In the following, we call non-bouncing tree a tree containing no bouncing cell.

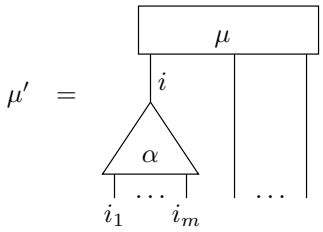
Lemma 2.3 *Let μ be a net.*

1. *Let $i_1, \ldots, i_m$ be free ports of μ, and τ a non-bouncing tree with m leaves. Then, if we pose*



we have $\mu'\Uparrow_i$ iff $\mu\Uparrow_{i_k}$ for all $1 \leq k \leq m$, and for any free port j which is free in both μ and μ', we have $\mu'\Downarrow_j$ iff $\mu\Downarrow_j$.

2. *Let i be a blind free port of μ. Then, if we pose*



we have $\mu' \Uparrow_{i_k}$ *for all* $1 \leq k \leq m$*, where* α *is an arbitrary cell of arity* m*, and for any port* j *which is free in both* μ *and* μ'*, we have* $\mu' \Downarrow_j$ *iff* $\mu \Downarrow_j$*.*

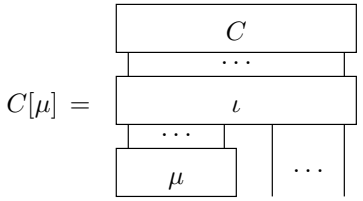
Proof. Part 1 is very easy. The fact that $\mu' \Uparrow_i$ iff $\mu \Uparrow_{i_k}$ for all k is a consequence of the hypothesis that τ is non-bouncing, and of the remark that all observable paths starting from i must pass through one of the i_k. The fact that $\mu' \Downarrow_j$ iff $\mu \Downarrow_j$ for $j \neq i$ is also trivial: τ hardly changes anything, since it does not interact with μ and is perfectly transparent to observable paths.

On the other hand, at the moment we are not able to show that part 2 holds; its proof requires the notion of *bisimilarity*, and is therefore deferred to Sect. 2.2.3 □

2.1.3 Observational equivalence

The observational equivalence we shall introduce will be a *contextual equivalence*, i.e., it will be based on observing the outcome of plugging a net into a *context*. In the framework of interaction nets, a context is particularly easy to define: it is just... a net, with a compatible interface. For convenience, we shall assume in the following that all nets have their free ports numbered by a positive integer from 1 to n, where n is the number of free ports.

Definition 2.6 (Context) *A* context *for nets with* n *free ports is a net* C *with at least* $n+1$ *free ports, together with an injection* ι *from* $\{1, \ldots, n\}$ *to the free ports of* C*. If* μ *is a net with* n *free ports, we denote by* $C[\mu]$ *the net obtained by plugging each free port* i *of* μ *to the free port* $\iota(i)$ *of* C*, i.e., graphically, we have*



where ι *is the wiring corresponding to the injection.*

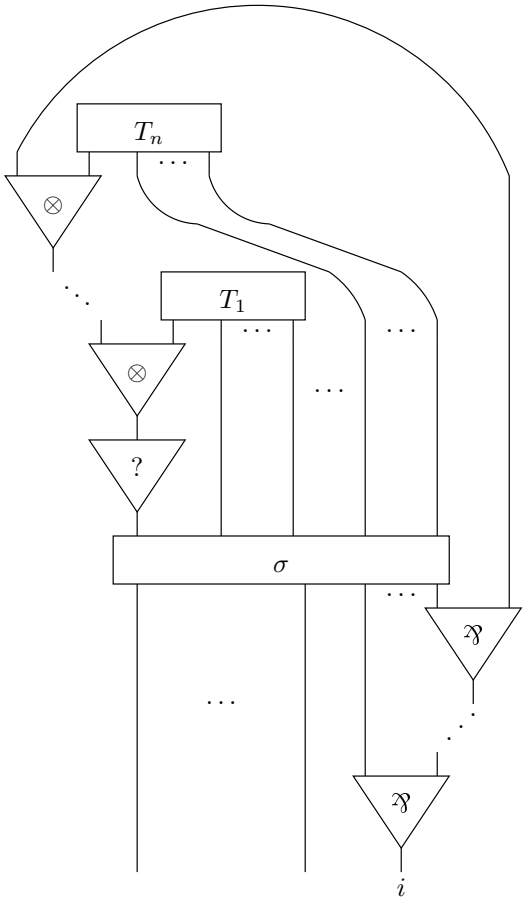
The reader may have remarked that the notation $C[\mu]$ "forgets" the injection ι; as a matter of fact, the rôle of ι is purely formal, and we shall nearly always leave it implicit in the sequel, i.e., when we say "for any context C..." we actually mean "for any net C with a compatible interface and any injection ι...", and when we say "there exists a context C..." we actually mean "there exists a net C with a compatible interface and an injection ι...". In fact, according to our convention, if the interface of μ is $I = \{1, \ldots, n\}$ and that of C

is $J = \{1, \ldots, n+k+1\}$, we can always consider ι to be the canonical injection of I into J, i.e., the one behaving like the identity. This particularly economical solution is the one we adopt in graphical representations.

We are now ready to define our notion of observational equivalence:

Definition 2.7 (Observational equivalence) *Let μ, ν be two nets with the same number of free ports. We say that μ and ν are* observationally equivalent*, and we write $\mu \simeq \nu$, iff, for any context C, $C[\mu]\Downarrow$ iff $C[\nu]\Downarrow$.*

This notion of observational equivalence can be seen as an extension of head-normalization equivalence in the λ-calculus, which is defined as follows: let T be a λ-term, and write $T\Downarrow$ iff T has a head normal form, or equivalently iff the head reduction of T terminates, or equivalently iff T is solvable. Then, two λ-terms T, U are observationally equivalent with respect to head normal form iff, for every context $C[\cdot]$, $C[T]\Downarrow$ iff $C[U]\Downarrow$. In fact, when we see λ-terms as multiplicative exponential linear logic proof-nets, or as sharing graphs (which are interaction nets), a head normal form $\lambda\vec{x}.xT_1 \ldots T_n$ looks something like



where $\otimes$ and $\parr$ cells represent resp. applications and abstractions (@ and λ nodes in sharing graphs), the ? cell is a *dereliction* corresponding the head variable (a *croissant* in sharing graphs), and σ is a net contracting/weakening variables (in case they appear more than once or not at all) and assigning them

to their respective abstractions. The free port labelled by i is the "root" of the term, while the other free ports represent free variables.

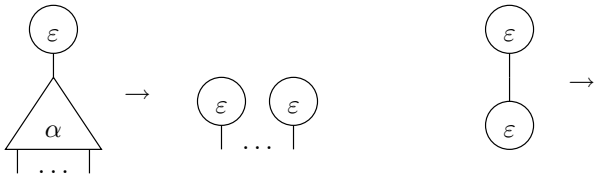
One clearly sees that, if we call μ the above net, then $\mu{\downarrow_i}$. Therefore, head-normalization equivalence in the λ-calculus coincides with our observational equivalence restricted to a distuinguished free port, the "root". The fact that λ-terms have a "root" is what allows one to define the very concept of head normal form. In interaction nets, no free port can in general be assigned a privileged rôle, so there may be several "head normal forms". Our observational equivalence cannot but treat all of them equally, without any of them being considered more "special" than any other.

The first thing we verify is that $\simeq_\beta \subseteq \simeq$, as one might reasonably expect considering the determinism of interaction nets reduction:

Proposition 2.4 *Let μ, ν be two nets such that $\mu \simeq_\beta \nu$. Then, $\mu \simeq \nu$.*

Proof. By definition, $\mu \simeq_\beta \nu$ means that there exists a net o such that $\mu \to^* o$ and $\nu \to^* o$. Then, for any suitable context C, we have $C[\mu] \to^* C[o]$ and $C[\nu] \to^* C[o]$, so obviously $C[\mu]\Downarrow$ iff $C[\nu]\Downarrow$. □

If the INS considered has an eraser cell, which we call ε, then not all contexts are actually needed to discriminate two nets with respect to $\simeq$. An eraser cell is a cell interacting as follows:

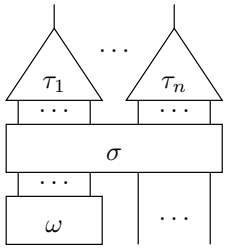


for all cells α of the system. Notice that eraser cells are non-bouncing; the principal port of an eraser cell can be seen as the "incarnation" of a blind port. An INS containing an eraser cell will be called an εINS.

Definition 2.8 (Principal context) *A context C, with associated injection ι, is* principal *iff:*

1. *it is cut-free;*
2. *if i is a free principal port of C, then it is in the image of ι;*
3. *if i, j are two free ports of C connected by a wire, then they are in the image of ι.*

In other words, a principal context has the following shape:

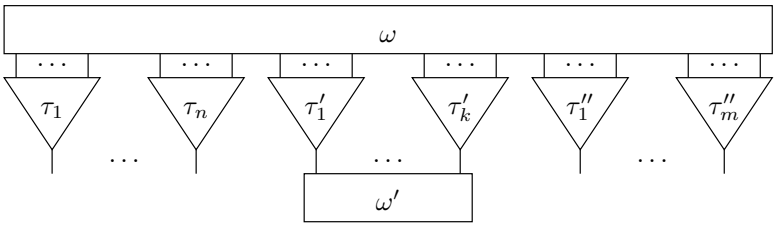


where the "upper" free ports are those connected to the net to which the context is applied.

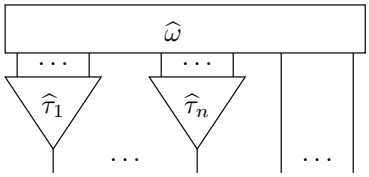
Lemma 2.5 (Context) *Let μ, ν be two nets of an εINS, with the same number of free ports. If $\mu \not\simeq \nu$, then there exists a principal context C discriminating between μ and ν, i.e., such that $C[\mu]\Downarrow$ and $C[\nu]\Uparrow$, or vice versa.*

Proof. By hypothesis, there exists a context C' such that, for example, $C'[\mu]\Downarrow$ and $C'[\nu]\Uparrow$. We shall gradually remove the features of C' making it non-principal, thus building a principal context C.

We can write the generic context C' as follows:

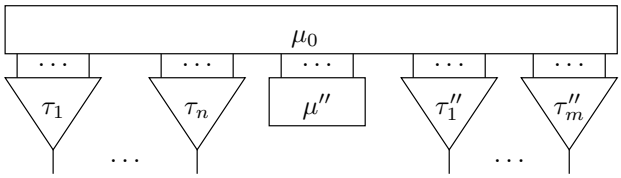


where the roots of the trees $\tau_1, \ldots, \tau_n$ are the free ports which will be connected to those of μ and ν when C' is applied to them. Our goal is to arrive to a context C of the form

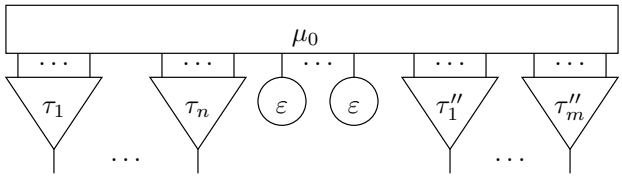


having the same behavior as C' with respect to μ and ν, and where no wire of $\widehat{\omega}$ connects two of the "rightmost" free ports.

Since $C'[\mu]\Downarrow$, by definition we have that $C'[\mu] \to^* \mu'$, where μ' is a net containing an immediately observable free port. In the reduction sequence leading from $C'[\mu]$ to μ', some of the active pairs reduced may be already present in C'. By strong confluence, we can choose a reduction sequence that reduces all and only these active pairs first, and then everything else. In other words, we split the reduction sequence into $C'[\mu] \to^* C_1[\mu] \to^* \mu'$, where C_1 is of the following shape:



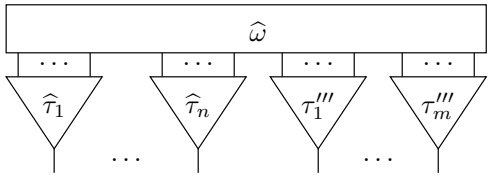
with μ_0 a suitable cut-free net. Now, by hypothesis, the subnet μ'' does not take part in the interaction with μ, so if we put

$C_2 =$ 

we obviously have $C_2[\mu]\Downarrow$. By looking at the interaction rules for ε, it is not hard to see that, since μ_0 is cut-free, C_2 has a cut-free form C_3, which of course, by confluence, verifies $C_3[\mu]\Downarrow$.

For what concerns ν, we also have $C'[\nu] \to^* C_1[\nu]$. Now, if, as we know by hypothesis, no observable path is ever created during the reduction of $C_1[\nu]$, then there can be no observable path created after replacing μ'' with ε cells, since the principal ports of eraser cells are the archetypal blind ports. Therefore, we have $C_2[\nu]\Uparrow$, and so $C_3[\nu]\Uparrow$.

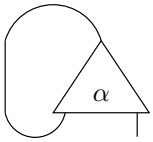
We have thus arrived at a discriminating context C_3 of the form



To conclude, we can assume that $\tau_1''', \ldots, \tau_m'''$ are non-bouncing, and that there is no connection between any of their leaves in $\widehat{\omega}$. In fact, such situations would yield observable paths whose presence is independent of the interaction with μ or ν, and which therefore add no discriminatory power to C_3. Then, by part 1 of Lemma 2.3, $\tau_1''', \ldots, \tau_m'''$ cannot hide nor add observable paths, so we can completely remove them, obtaining our principal context C. □

Notice that eraser cells are not used to actually erase, but rather to allow the restriction to cut-free contexts. In fact, as far as observational equivalence is concerned, an eraser cell ought to be seen as a "reification of blindness": it is a way to represent a blind port without resorting to vicious circles or infintely-reducing nets. It is fair to say that erasers play the same rôle as Ω leaves in Bhöm trees. In fact, any blind net with n free ports is, by definition, observationally equivalent to a net consisting of n ε cells.

If erasers are not available, a similar effect would be obtained by a configuration like

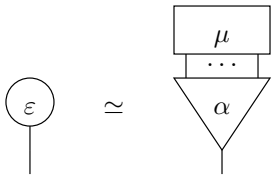


where α is a binary cell. Of course, in this case principal contexts would loose the property of being cut-free. Anyway, since eraser cells arise very naturally when programming with interaction nets, for practical purposes one can always consider INS's to have eraser cells.

It may be interesting to analyze the relationship between our observational equivalence and that defined by Fernández and Mackie [FM03]. We do not have precise results in this sense, although it seems clear that the two are quite orthogonal, i.e., neither is included in the other. In fact, Fernández&Mackie's equivalence is in some sense more "intensional". It allows for example equivalences like

whenever eraser cells only appear in situations in which a net reduces to the empty net, i.e., it is actually erased; in that case, the equivalence (seen from right to left) corresponds to a natural optimization which speeds up the erasing process, and is therefore perfectly sound. By contrast, our observational equivalence is crucially based on the distinction between the two nets above, which are resp. the prototypes of a blind and an observable net.

On the other hand, our observational equivalence makes the identification



whenever μ is blind and α is non-bouncing, whereas Fernández&Mackie's equivalence distinguishes between the two nets if α is a constructor[1], and nothing forbids a constructor to be non-bouncing (the λ node in sharing graphs is a typical example). This last distinction appears much like the distinction between $\lambda x.\Omega$ and Ω in the observational equivalence based upon weak-head-normalization in the λ-calculus. In fact, Fernández&Mackie's equivalence, which uses the concept of *interface normal form*, may be closer to observing weak head normal forms rather than head normal forms.

2.2 Bisimilarity

In this section we develop a notion of *bisimulation*, with the associated *bisimilarity*, which may be of help in proving observational equivalence of nets. This is similar to what done by Fernández and Mackie [FM03], although our definition of observational equivalence (and above all of bisimilarity) is different, as discussed above.

The bisimilarity we define is inspired to what is usually called *barbed bisimilarity* in process calculi (cf. Sect. 5.1, and Sangiorgi and Walker's book for a complete reference [SW01]). Actually, we shall see that the confluence property of interaction nets reduction in some sense trivializes this notion with respect to the concurrent case. Nevertheless, it will turn out to be crucial in proving several results, and it will be the source of inspiration for a similar definition, this time in a non-confluent framework, that will be needed in the multiport case to speak of behavioral equivalence (Sect. 6.3).

As done above, all throughout the rest of the section we fix a generic untyped INS $(\Sigma, \mathcal{R})$, and we take the terms "cell", "net", and "rule" to implicitly mean resp. "cell of Σ", "net of $\langle\Sigma\rangle$", and "rule of $\mathcal{R}$".

[1] Fernández&Mackie's equivalence is based among other things upon a partitioning of the alphabet of an INS into *constructors* and *destructors*, which is specified when giving the alphabet itself (as in [Laf90]). Their observational equivalence takes into account the constructor/destructor nature of cells, which we do not do. This is another example of the "intensionality" of Fernández&Mackie's equivalence.

2.2.1 Bisimulation and bisimilarity

In what follows, we use the notation $\mu\downarrow$ to say that there exists a free port of μ which is immediately observable.

Definition 2.9 (Bisimulation) *A binary relation $\mathcal{B}$ relating only nets with the same number of free ports is a* bisimulation *iff, whenever $(\mu, \nu) \in \mathcal{B}$, we have:*

1. *$\mu\downarrow$ implies $\nu\Downarrow$;*
2. *$\mu \to \mu'$ implies that there exists ν' such that $\nu \to^* \nu'$ and $(\mu', \nu') \in \mathcal{B}$;*
3. *$\nu\downarrow$ implies $\mu\Downarrow$;*
4. *$\nu \to \nu'$ implies that there exists μ' such that $\mu \to^* \mu'$ and $(\mu', \nu') \in \mathcal{B}$.*

Immediate observability and single-step reduction can actually be replaced resp. by observability and reduction:

Lemma 2.6 *Let $\mathcal{B}$ be a bisimulation, and let $(\mu, \nu) \in \mathcal{B}$. Then:*

1. *$\mu\Downarrow$ iff $\nu\Downarrow$;*
2. *if $\mu \to^* \mu'$, then $\nu \to^* \nu'$ such that $(\mu', \nu') \in \mathcal{B}$, and similarly with the rôles of μ and ν exchanged.*

Proof. We start with the second statement. If $\mu \to^* \mu'$, by definition $\mu \to \mu_1 \to \cdots \to \mu_{n-1} \to \mu_n = \mu'$; we reason by induction on n. If $n = 0$, then $\mu' = \mu$, and the fact is vacuously true. If $n > 0$, then we know by induction hypothesis that there exists ν'' such that $\nu \to^* \nu''$ and $(\mu_{n-1}, \nu'') \in \mathcal{B}$. But $\mathcal{B}$ is a bisimulation, so $\mu_{n-1} \to \mu'$ implies the existence of a ν' such that $\nu'' \to^* \nu'$ and $(\mu', \nu') \in \mathcal{B}$, which is what we were looking for. In case $\nu \to^* \nu'$, finding a μ' such that $\mu \to^* \mu'$ and $(\mu', \nu') \in \mathcal{B}$ is done following exactly the same argument.

Suppose now that $\mu\Downarrow$. By definition, we have $\mu \to^* \mu'\downarrow$. By point 2 just proved, we have $\nu \to^* \nu'$ with $(\mu', \nu') \in \mathcal{B}$; but since $\mathcal{B}$ is a bisimulation, $\mu'\downarrow$ implies $\nu'\Downarrow$. Now, since ν' is a reduct of ν, we also have $\nu\Downarrow$. The proof of the converse is perfectly analogous, just exchanging the rôles of μ and ν. □

The standard properties of bisimulations are verified; they are straightforward consequences of the definition:

Lemma 2.7 *Bisimulations are closed under inversion, arbitrary unions, and reflexive-transitive closure.*

Proof. The proofs of all three properties are standard and left to the reader. Lemma 2.6 is needed to prove closure under transitivity. □

Definition 2.10 (Bisimilarity) Bisimilarity*, denoted $\dot{\sim}$, is the union of all bisimulations.*

Notice that, by definition, two nets μ, ν are bisimilar iff there exists a bisimulation $\mathcal{B}$ such that $(\mu, \nu) \in \mathcal{B}$. From Lemma 2.7 it follows that $\dot{\sim}$ is an equivalence relation, and that it is also a bisimulation; in particular, we have

Lemma 2.8 *Bisimilarity is the largest equivalence relation such that, whenever* $\mu \dot{\sim} \nu$*, we have*

1. $\mu\Downarrow$ *iff* $\nu\Downarrow$*;*
2. *if* $\mu \to^* \mu'$*, then* $\nu \to^* \dot{\sim} \mu'$*.*

Proof. As observed above, this is a straight-forward consequence of Lemmas 2.6 and 2.7, and, for what concerns maximality, of the fact that bisimilarity is defined to be the union of all bisimulations. □

Just as with observational equivalence, the first result we prove about bisimilarity is that it contains β-equivalence.

Lemma 2.9 *Let* μ, ν *be two nets such that* $\mu \simeq_\beta \nu$*. Then,* $\mu \dot{\sim} \nu$*.*

Proof. It is enough to prove that $\simeq_\beta$ is a bisimulation. By definition, $\mu \simeq_\beta \nu$ means that there exists o such that $\mu \to^* o$ and $\nu \to^* o$. Now suppose $\mu\downarrow$. Observable paths are preserved under reduction, so $o\downarrow$; but o is also a reduct of ν, so $\nu\Downarrow$. Now suppose that $\mu \to \mu'$; this implies in particular that $\mu \simeq_\beta \mu'$, which by transitivity of $\simeq_\beta$ implies $\mu' \simeq_\beta \nu$. Points (3) and (4) of Definition 2.9 are established in a similar way. □

Observe how, in the above lemma, the transitivity of $\simeq_\beta$, which is a consequence of the confluence of reduction, makes the proof of properties (2) and (4) of Definition 2.9 completely trivial. In fact, the Church-Rosser property implies a trivialization of the whole definition of bisimilarity, namely that the latter is characterized by property 1 of Lemma 2.8:

Lemma 2.10 $\mu \dot{\sim} \nu$ *iff* μ *and* ν *are either both observable, or both blind.*

Proof. The forward implication is nothing but point 1 of Lemma 2.8. For the converse, it is enough to show that the set

$$\mathcal{B} = \{(\mu, \nu) \mid \mu\Downarrow \text{ iff } \nu\Downarrow\}$$

is a bisimulation. Properties (1) and (3) of bisimulations hold by definition of $\mathcal{B}$; if $\mu \to \mu'$, by confluence we have $\mu'\Downarrow$ iff $\mu\Downarrow$, but by definition of $\mathcal{B}$ $\mu\Downarrow$ iff $\nu\Downarrow$, so $(\mu', \nu) \in \mathcal{B}$. The same can be said in case $\nu \to \nu'$, so $\mathcal{B}$ is indeed a bisimulation. □

Corollary 2.11 $\mu \simeq \nu$ *iff* $C[\mu] \dot{\sim} C[\nu]$ *for every context* C*.*

The above result allows us to prove observational equivalence using coinductive techniques, similarly to what is usually done in process calculi, and to what Fernández and Mackie have already done in the context of interaction nets [FM03]. The advantage is that, instead of having to prove $C[\mu]\Downarrow$ iff $C[\nu]\Downarrow$, we can first show that the weaker implications $C[\mu]\downarrow \Rightarrow C[\nu]\Downarrow$ and $C[\nu]\downarrow \Rightarrow C[\mu]\Downarrow$ hold, and then show that $C[\mu]$ and $C[\nu]$ are able to simulate the one-step reductions of each other. Of course there is still the quantification over all contexts which may make the task difficult, but in several interesting cases (see Sect. 2.3.1) Corollary 2.11 will turn out be just what we need.

2.2.2 Bisimulations up to reflexive-transitivity

We now introduce the notion of *bisimulation up to reflexive-transitivity*, which can be very useful in proving that a relation which is the reflexive-transitive closure of another relation is a bisimulation. The reader can safely skip this section until the notion is used in Sect. 2.3.1.

Definition 2.11 (Bisimulation up to reflexive-transitivity) *Let $\mathcal{B}$ be a binary relation relating only nets with the same number of free ports. We say that $\mathcal{B}$ is a* bisimulation up to reflexive-transitivity *iff, whenever $(\mu, \nu) \in \mathcal{B}$, we have:*

1. *$\mu\downarrow$ implies $\nu\Downarrow$;*
2. *$\mu \to \mu'$ implies that there exists ν' such that $\nu \to^* \nu'$ and $(\mu', \nu') \in \mathcal{B}^*$;*
3. *$\nu\downarrow$ implies $\mu\Downarrow$;*
4. *$\nu \to \nu'$ implies that there exists μ' such that $\mu \to^* \mu'$ and $(\mu', \nu') \in \mathcal{B}^*$;*

where $\mathcal{B}^*$ denotes the reflexive-transitive closure of $\mathcal{B}$.

The reader can check that the proof of Lemma 2.6 can be easily adapted *mutatis mutandi* to bisimulations up to reflexive-transitivity: if $(\mu, \nu) \in \mathcal{B}$ for a bisimulation up to reflexive-transitivity $\mathcal{B}$, then $\mu\Downarrow$ iff $\nu\Downarrow$, and $\mu \to^* \mu'$ implies that there exists ν' such that $\nu \to^* \nu'$ with $(\mu', \nu') \in \mathcal{B}^*$, and vice versa exchanging the rôles of μ and ν.

Of course, if $\mathcal{B}$ is a bisimulation up to reflexive-transitivity, then it is not in general a bisimulation. However, $\mathcal{B}^*$ is:

Lemma 2.12 *If $\mathcal{B}$ is a bisimulation up to reflexive-transitivity, then $\mathcal{B}^*$ is a bisimulation.*

Proof. Let $(\mu, \nu) \in \mathcal{B}^*$. If $\mu = \nu$, the four properties defining bisimulations (Definition 2.9) hold trivially. If $\mu \neq \nu$, then by definition there exist n nets $o_1, \ldots, o_n$ such that $(\mu, o_1), (o_1, o_2), \ldots, (o_{n-1}, o_n), (o_n, \nu) \in \mathcal{B}$. We shall prove that properties (1) through (4) of Definition 2.9 hold by induction on n. If $n = 0$, this is true thanks to the hypothesis that $\mathcal{B}$ is a bisimulation up to reflexive-transitivity. If $n > 0$, let us pose $o_n = o$. By induction hypothesis, properties (1) through (4) hold for the pair (μ, o). These properties have the consequence (cf. Lemma 2.6) that $\mu\Downarrow$ iff $o\Downarrow$, and, as remarked above, since $\mathcal{B}$ is a bisimulation up to reflexive-transitivity and $(o, \nu) \in \mathcal{B}$, we have that $o\Downarrow$ iff $\nu\Downarrow$, so properties (1) and (3) are verified for (μ, ν). Now suppose that $\mu \to \mu'$. By induction hypothesis, $o \to^* o'$ with $(\mu', o') \in \mathcal{B}^*$; by property (2) of Lemma 2.6, and because $(o, \nu) \in \mathcal{B}$ with $\mathcal{B}$ bisimulation up to reflexive-transitivity, $o \to^* o'$ implies $\nu \to^* \nu'$ such that $(o', \nu') \in \mathcal{B}^*$. We then conclude that $(\mu', \nu') \in \mathcal{B}^*$ by transitivity of $\mathcal{B}^*$. This proves property (2); the same argument applies to show that property (4) holds as well. □

2.2.3 Proof of part 2 of Lemma 2.3

To close this section on bisimilarity, we give a direct application of the concepts introduced so far, or more precisely to a variant of them. In fact, we shall give a proof of part 2 of Lemma 2.3 using a notion which is related to, but stronger than, bisimilarity:

Definition 2.12 (Portwise bisimulation) *A binary relation $\mathcal{B}$ relating only nets with the same number of free ports is a* portwise bisimulation *iff, whenever $(\mu, \nu) \in \mathcal{B}$, we have:*

1. *$\mu\downarrow_i$ implies $\nu\Downarrow_i$ for every free port i;*
2. *$\mu \to \mu'$ implies that there exists ν' such that $\nu \to^* \nu'$ and $(\mu', \nu') \in \mathcal{B}$;*
3. *$\nu\downarrow_i$ implies $\mu\Downarrow_i$ for every free port i;*
4. *$\nu \to \nu'$ implies that there exists μ' such that $\mu \to^* \mu'$ and $(\mu', \nu') \in \mathcal{B}$.*

Notice that in points (1) and (3) it is sound to quantify over "all free ports" because we have supposed that μ and ν have the same interface.

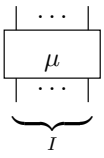
It is not hard to check that all of the properties verified by bisimulations, namely Lemmas 2.6 and 2.7, are verified by portwise bisimulations too. Therefore, we can define *portwise bisimilarity* as expected:

Definition 2.13 (Portwise bisimilarity) Portwise bisimilarity*, denoted $\dot{\approx}$, is the union of all portwise bisimulations.*

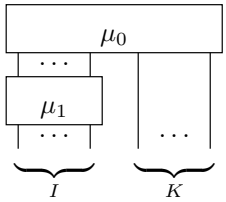
An adapted version of Lemma 2.8 holds for portwise bisimilarity; in particular, what is interesting to us is that $\mu \dot{\approx} \nu$ implies $\mu\Downarrow_i$ iff $\nu\Downarrow_i$ for all free ports i of μ and ν.

In the sequel, we shall assume that the INS we are considering has an eraser cell; if it does not, we just add it to it, and our arguments will not be harmed.

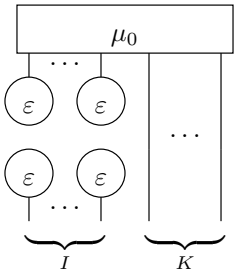
Let μ be a net, and I a subset of its interface. We shall say that μ is *relatively blind on I* iff, for all μ' such that $\mu \to^* \mu'$, there is no observable path in μ' between any two free ports (not necessarily distinct) of I. In what follows, we shall always represent a relatively blind net μ with its relatively blind interface I drawn at the bottom, and the rest of the interface drawn at the top:



If μ_0 is a net with interface $J \cup K$ such that all of the free ports in J are blind, and if μ_1 is a net with interface $I \cup J'$ which is relatively blind on I and such that J' is in bijection with J, then we shall write $\mu_0 \bullet \mu_1$ for the net obtained by plugging the free ports in J with the free ports in J' according to the bijection. In general, we shall assume the bijection to be the identity, and picture the situation as follows:



If $\mu = \mu_0 \bullet \mu_1$ as above, we shall write μ_ε for the net obtained by replacing μ_1 with a suitable net made entirely of eraser cells:



We shall see that $\mu \mathrel{\dot{\approx}} \mu_\varepsilon$, so the free ports of I are actually blind in μ.

In fact, define the set

$$\mathcal{B} = \{(\mu, \mu_\varepsilon) \mid \text{for all } \mu \text{ of the form } \mu_0 \bullet \mu_1\}.$$

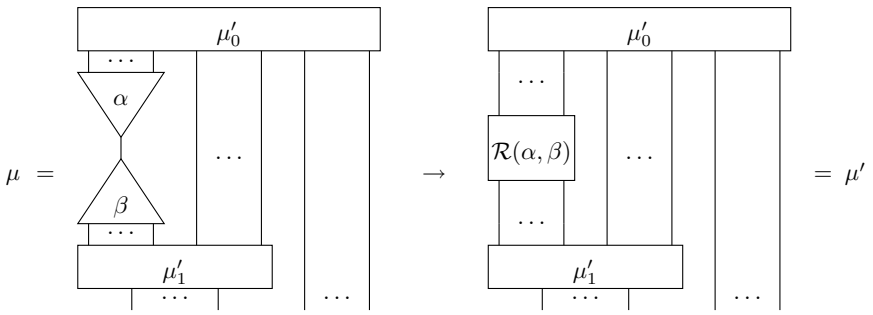
Then, we have the following:

Claim 2.3.1 *$\mathcal{B}$ is a portwise bisimulation.*

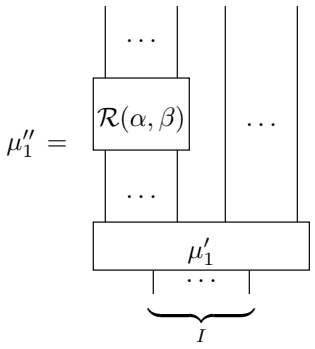
Proof. In what follows, I, J, K denote sets of free ports as in the above definitions and pictures. Let us start with property (3) of Definition 2.12. If $\mu_\varepsilon{\downarrow_k}$, then clearly $k \in K$, because all free ports of I are blind in μ_ε. Then, the observable path must obviously pass through μ_0, and thus $\mu{\downarrow_k}$.

Similarly, if we suppose $\mu{\downarrow_k}$, by the hypothesis that μ_1 is relatively blind on I, assuming $k \in I$ would imply that there is an observable path entering μ_0 through a port of J; but sub-paths of observable paths are observable (once we consider their extremities as free ports), so this would imply $\mu_0{\downarrow_j}$ for some $j \in J$, a contradiction, since all ports of J are blind by hypothesis. Therefore, $k \in K$, and so $\mu_\varepsilon{\downarrow_k}$, which proves property (1).

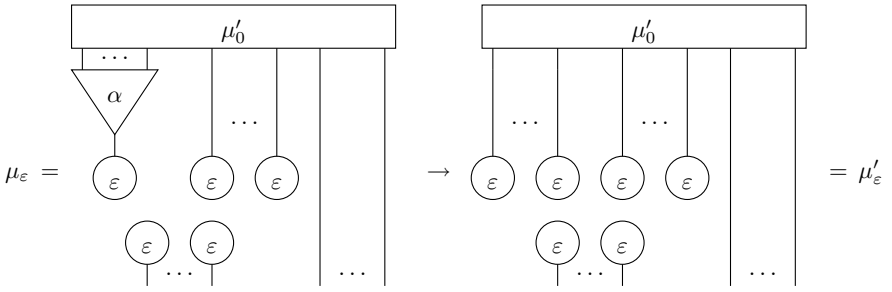
Let us turn to property (2), for which we suppose that $\mu \to \mu'$. If the reduction takes place inside μ_0, i.e., $\mu_0 \to \mu_0'$, then μ_0' is also blind on J by definition, so $\mu' = \mu_0' \bullet \mu_1$; but clearly $\mu_\varepsilon \to \mu_\varepsilon'$, and by definition $(\mu', \mu_\varepsilon') \in \mathcal{B}$. Another possibility is that the reduction takes place inside μ_1, i.e., $\mu_1 \to \mu_1'$. By definition, μ_1' is also relatively blind on I, so $\mu' = \mu_0 \bullet \mu_1'$, and since the interface of μ_1' is the same as that of μ_1, we have $\mu_\varepsilon' = \mu_\varepsilon$, so $(\mu', \mu_\varepsilon) \in \mathcal{B}$. The last possibility is that of an interaction between a cell of μ_0 and a cell of μ_1, in which case we have



We start by observing that α is not bouncing, otherwise there would be an immediately observable port in J, contradicting our hypotheses; then, by part 1 of Lemma 2.3, the "leftmost" ports of μ_0' are all blind. Now pose

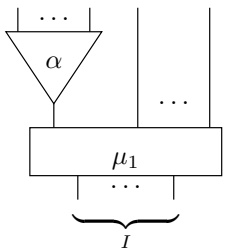


and suppose that a reduct of μ_1'' has an observable path ϕ between two ports of I. Notice that, since β does not participate in any interaction of μ_1, removing it hardly changes anything, so μ_1' is also relatively blind on I. Therefore, ϕ must pass through (a reduct of) $\mathcal{R}(\alpha, \beta)$. Then, observable paths being particular interaction paths, by Proposition 2.1 there would be an interaction path between two "lower" ports of $\mathcal{R}(\alpha, \beta)$ (i.e., those ports which were connected to the auxiliary ports of β in μ). But $\mathcal{R}(\alpha, \beta)$ is cut-free, so such a path would actually be observable, contradicting the fact that α is non-bouncing. This proves that μ_1'' is relatively blind on I, and that $\mu' = \mu_0' \bullet \mu_1''$. Now, since ε are eraser cells, in μ_ε we have



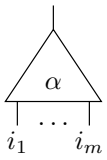
and $(\mu', \mu_\varepsilon') \in \mathcal{B}$ by definition of $\mathcal{B}$.

We have only property (4) left to prove. For this, suppose $\mu_\varepsilon \rightarrow \mu_\varepsilon'$. Again, if the reduction takes place inside μ_0, the statement is trivial. Otherwise, the only other possibility is that an eraser cell interacts with μ_0 as in the above picture. For what concerns μ, suppose that $\mu_1 \rightarrow^* \mu_1'$ such that the free port of μ_1' connected to the cell α is principal; then we are back to the above case, and the result follows by similar considerations. Otherwise, the port connected to α is not principal in all reducts of μ_1. Then, α can never interact with μ_1, hence the net

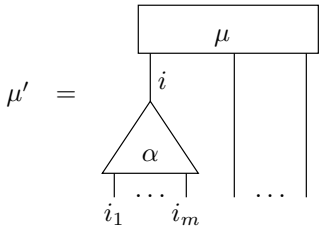


is relatively blind on I, and $(\mu, \mu'_\varepsilon) \in \mathcal{B}$ by definition. □

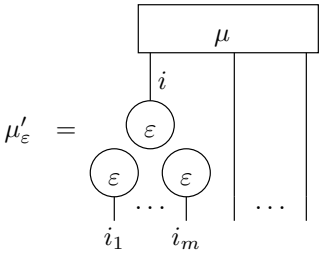
Part 2 of Lemma 2.3 is a consequence of the above claim. In fact, notice that, for any cell α of arity m, the net



is relatively blind on $\{i_1, \ldots, i_m\}$. Then, if i is a blind free port of a net μ, and if



then by Claim 2.3.1 we have $\mu' \dot{\approx} \mu'_\varepsilon$, where



Therefore, $\mu' \Uparrow_{i_l}$ for all $1 \leq l \leq m$; additionally, for any port k which is free in both μ and μ', we have $\mu' \Downarrow_k$ iff $\mu'_\varepsilon \Downarrow_k$ iff $\mu \Downarrow_k$. The last equivalence comes from the fact that an observable path in a reduct of μ'_ε cannot use the ε cell connected to the port i (or any of the ε cells it may generate), hence such a path must exist in a reduct of μ as well.

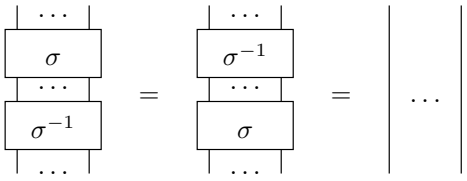
2.3 Path-based equivalence in the interaction combinators

In the following, we shall only consider the INS of the interaction combinators (see Sect. 1.3.2), so the terms "cell", "net", and "rule" will mean resp. "combinator", "net of combinators", and "combinator rule". Before getting to the heart of the matter, we make some preliminary definitions and conventions.

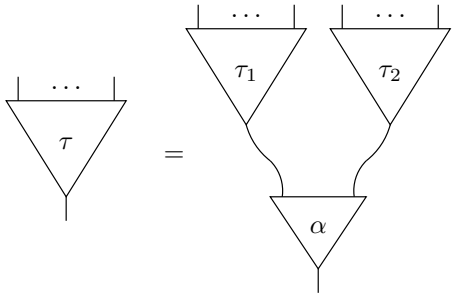
ε-wirings and inverse wirings. Together with the usual wirings, we shall also consider *ε-wirings*. These are wirings in which free ports are allowed to belong to ε cells. The following are examples of ε-wirings:

ε-wirings of the kind shown on the left will be denoted by $\widetilde{\sigma}$, those of the kind shown on the right will be denoted by $\widetilde{\omega}$.

Notice also that the wirings we denote with σ (cf. p. 21) can be considered permutations; given one such wiring, we can then define σ^{-1} to be the unique wiring such that



Trees. Contrarily to what we have done in the previous sections, in the case of the interaction combinators trees are considered to contain no ε cell. In other words, trees of combinators are defined inductively as follows: a single wire is a tree with one leaf, denoted by **1** (it is arbitrary which of the two extremities is the root and which is the leaf); if τ_1 and τ_2 are two trees with resp. n_1 and n_2 leaves, then the net



is a tree with $n_1 + n_2$ leaves, where $\alpha \in \{\gamma, \delta\}$. This tree is denoted $\alpha(\tau_1, \tau_2)$. Trees containing also ε cells will be called *tests* (see below).

Any tree can be annihilated my means of another tree:

Definition 2.14 (Cotree) *If τ is a tree, we define its* cotree $\tau^\dagger$ *by induction on τ:*

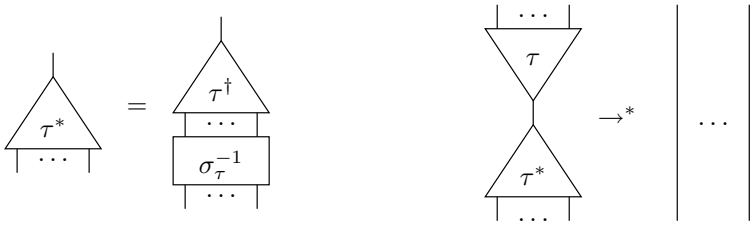
- $\mathbf{1}^\dagger = \mathbf{1}$*;*
- $\gamma(\tau_1, \tau_2)^\dagger = \gamma(\tau_2^\dagger, \tau_1^\dagger)$*;*
- $\delta(\tau_1, \tau_2)^\dagger = \delta(\tau_1^\dagger, \tau_2^\dagger)$.

It follows straight-forwardly from the definition that the co-cotree of τ is τ itself.

Lemma 2.13 (Cotree) *For any tree τ, the net obtained by plugging together τ and $\tau^\dagger$ through their roots reduces to a wiring.*

PROOF. By induction on τ, using the annihilation rules for γ and δ cells. $\square$

If we call σ_τ the wiring of Lemma 2.13, we can define a net τ^* which annihilates with τ in the simplest way, i.e., which yields the identity permutation:



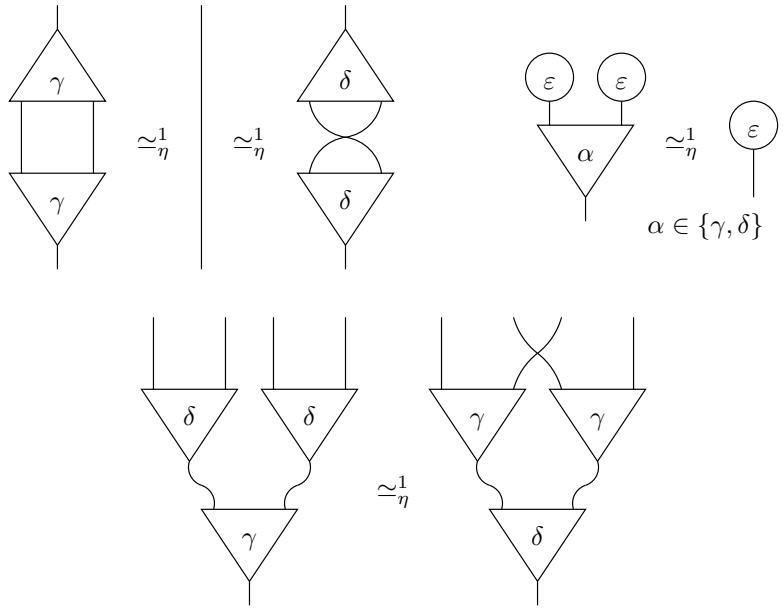
We shall say that any "tree plus permutation" τ' annihilating with τ as above is an *anti-tree* of τ. The net τ^* just defined is called the *canonical anti-tree* of τ.

Tests, ε-packages, filiform trees. We define an n-test to be a tree (in the above sense) with $n + m$ leaves such that m leaves are connected to principal ports of ε cells. Both m and n can be zero, but not at the same time. This means that trees with n leaves are special cases of n-tests. A *test* is simply an n-test for some n that we do not want or need to specify. Of particular interest to us will be 0-tests, 1-tests, and 2-tests; the first two will be called resp. *ε-packages* and *filiform trees*.

2.3.1 η- and $\beta\eta$-equivalence

We introduce a few additional rewriting rules to the system of the interaction combinators, which are not interaction rules but which will turn out to have very interesting properties with respect to observational equivalence.

Definition 2.15 (η-equivalence) *Given two nets μ, μ', we write $\mu \simeq^1_\eta \mu'$ iff they can be rewritten one into the other by means of exactly one of the following equalities, applied under any context:*



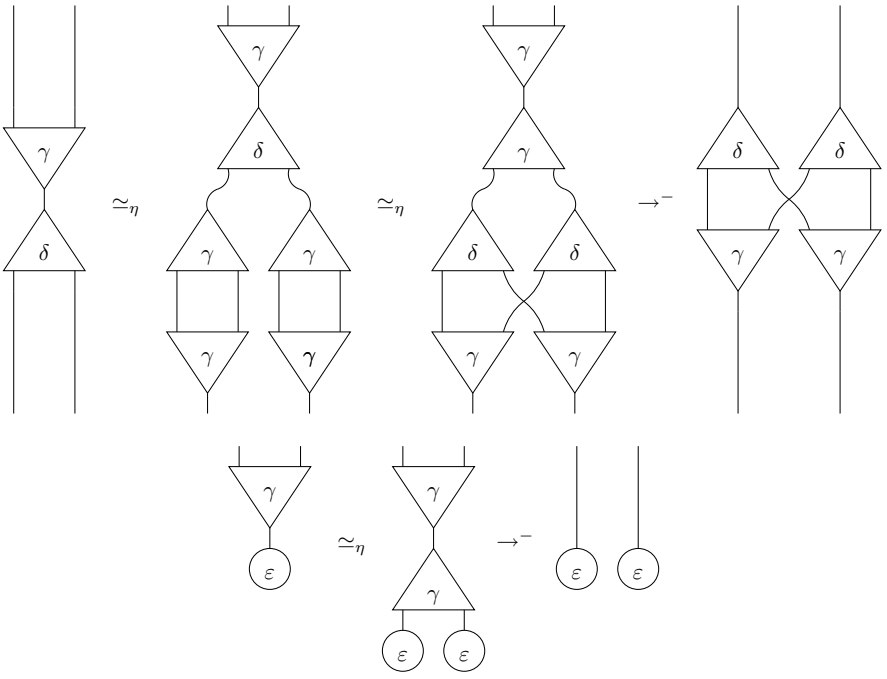
We write $\simeq_\eta$ for the reflexive-transitive closure of $\simeq_\eta^1$, and whenever $\mu \simeq_\eta \mu'$, we say that μ and μ' are η-equivalent*.*

The top-right and bottom equations of Definition 2.15, which we call resp. $\gamma\varepsilon$ (or $\delta\varepsilon$) and $\gamma\delta$ equations, were already considered by Lafont [Laf97]; in particular, the $\gamma\varepsilon$ and $\delta\varepsilon$ equations state the η-equivalence of all ε-packages to the ε combinator. On the other hand, the top-left equations, which we refer to as $\gamma\gamma$ and $\delta\delta$ equations, can be found in the work of Fernández and Mackie as part of a larger study on operational equivalence for interaction nets [FM03].

Definition 2.16 ($\beta\eta$-equivalence) *We write $\simeq_{\beta\eta}$ for the transitive closure of $\simeq_\beta \cup \simeq_\eta$, and if $\mu \simeq_{\beta\eta} \mu'$, we say that μ and μ' are* $\beta\eta$-equivalent*.*

Notice that, by definition, η- and $\beta\eta$-equivalence are actually congruences, i.e., if $\mu \simeq_\eta \nu$ (resp. $\mu \simeq_{\beta\eta} \nu$), then $C[\mu] \simeq_\eta C[\nu]$ (resp. $C[\mu] \simeq_{\beta\eta} C[\nu]$) for all contexts.

It is perhaps interesting to remark that $\beta\eta$-equivalence can actually be defined using a much smaller equivalence than $\simeq_\beta$. Let $\mu \to^- \mu'$ iff $\mu \to^* \mu'$ by means of annihilation rules only; of course $\to^-$ is confluent, so we can define an equivalence relation $\simeq_\beta^-$ as $\mu \simeq_\beta^- \nu$ iff there exists o such that $\mu \to^- o$ and $\nu \to^- o$. One can then prove that $\beta\eta$-equivalence is equal to the transitive closure of $\simeq_\beta^- \cup \simeq_\eta$. In other words, once η-equivalence is considered, the same equational theory is generated even if only half of the reduction rules are allowed. We show this in the case of the $\gamma\delta$ and the $\gamma\varepsilon$ commutations, the $\delta\varepsilon$ commutation being identical to the latter:



We also point out that there is no reasonable orientation for the equations defining η-equivalence, so there are no *canonical representatives* for the equivalence classes of $\simeq_{\beta\eta}$ on total nets (as opposed to $\beta\eta$-normal forms in the

λ-calculus). Nevertheless, we shall see that, just like in the λ-calculus, $\beta\eta$-equivalence is maximal on total nets: if two non-$\beta\eta$-equivalent total nets are identified, then all total nets must be identified (cf. Theorem 2.27).

Before that, we shall prove in the reminder of the section that if two nets are $\beta\eta$-equivalent, then they are also observationally equivalent. To do this, we apply the notion of bisimulation up to reflexive-transitivity, defined in Sect. 2.2.2.

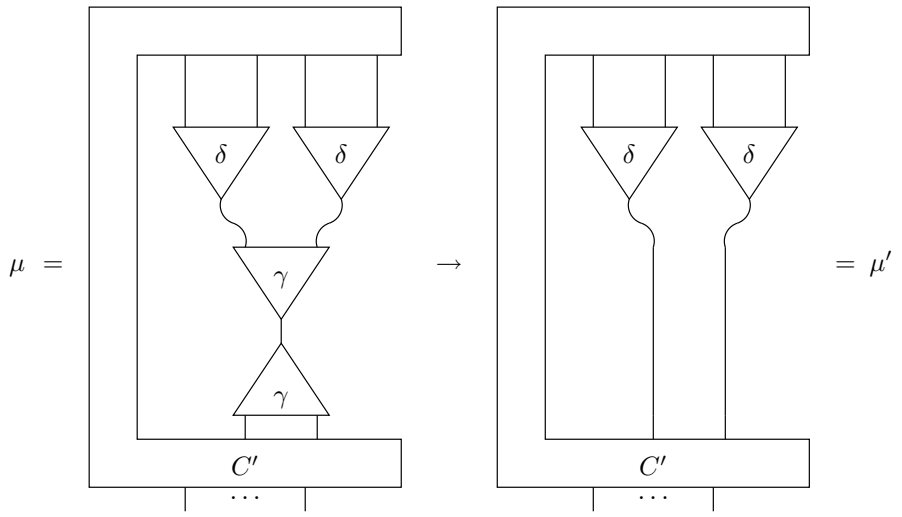
Lemma 2.14 *$\simeq_{\beta\eta}$ is a bisimulation.*

Proof. Observe that $\simeq_{\beta\eta}$ can be defined as the reflexive-transitive closure of the set $\mathcal{B} = \simeq_\beta \cup \simeq^1_\eta$. Therefore, by Lemma 2.12, it is enough to prove that $\mathcal{B}$ is a bisimulation up to reflexive-transitivity. Moreover, by Lemma 2.9, we know that $\simeq_\beta$ is a bisimulation, so the only thing that is left to prove is that, whenever $\mu \simeq^1_\eta \nu$, we have

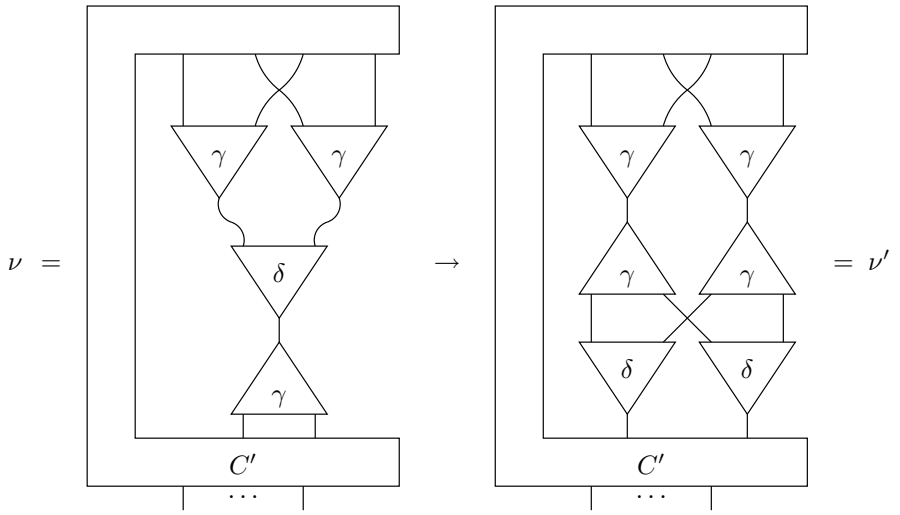
1. $\mu\downarrow$ implies $\nu\Downarrow$;
2. $\mu \to \mu'$ implies that there exists ν' such that $\nu \to^* \nu'$ and $\mu' \simeq_{\beta\eta} \nu'$;

and similarly for properties (3) and (4), which are obtained by exchanging the rôles of μ and ν in resp. (1) and (2). There is a total of five cases to be checked, one for each η-equation, but since the $\delta\delta$ and $\delta\varepsilon$ equations are virtually identical to the $\gamma\gamma$ and $\gamma\varepsilon$ equations, we shall neglect the former two equations and only analyze three cases.

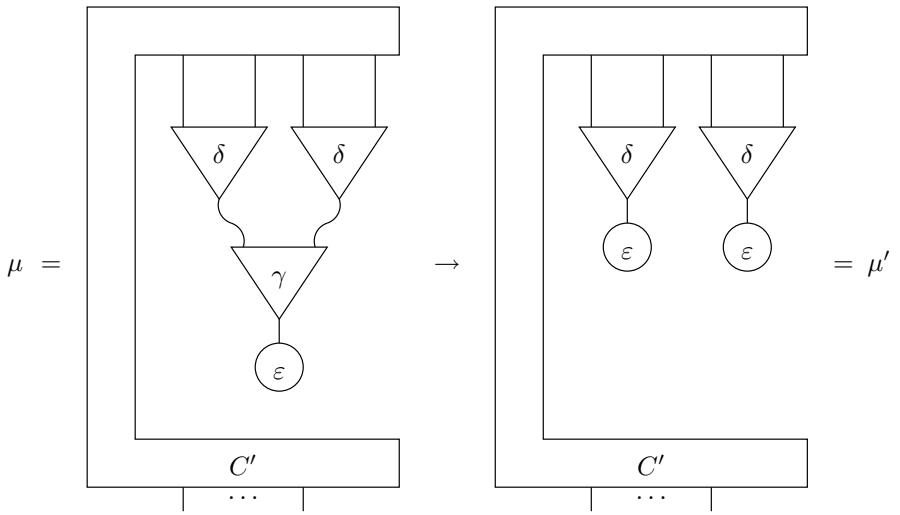
We start with the $\gamma\delta$ equation. Call μ_0 and ν_0 the two members of such equation, so that $\mu = C[\mu_0]$ and $\nu = C[\nu_0]$ for some context C. First of all, observe that the rewriting rule does not alter the presence of observable paths, so $\mu\downarrow$ iff $\nu\downarrow$, which is enough to establish properties (1) and (3). Now suppose $\mu \to \mu'$. If the reduction takes place inside C, i.e., $C[\mu_0] \to C'[\mu_0]$, then obviously $\nu = C[\nu_0] \to C'[\nu_0] \simeq^1_\eta C'[\mu_0]$. Otherwise, a cell has interacted with μ_0. Since μ_0 has only one principal port, there are three cases, one for each combinator. Let us start with the case of the γ combinator:



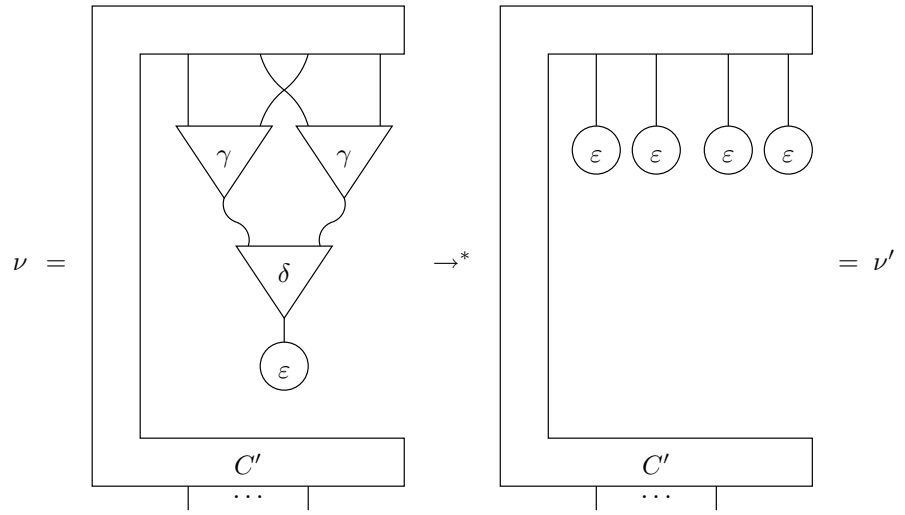
For what concerns ν, we can write



and the reader can check that $\nu' \rightarrow^* \mu'$, so in particular $\mu' \simeq_\beta \nu'$, which implies $\mu' \simeq_{\beta\eta} \nu'$. The case of the δ combinator is practically identical, so we pass directly to consider the ε combinator. We have

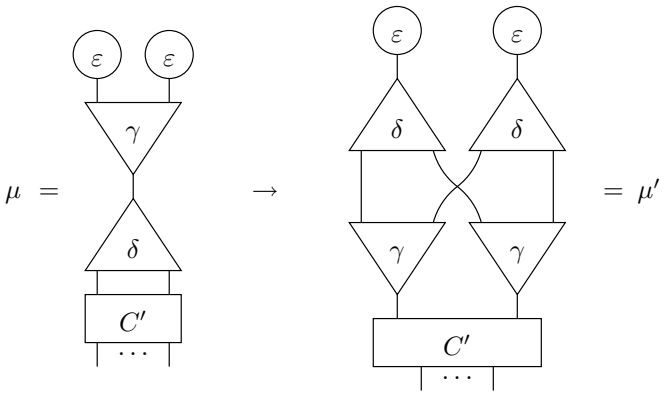


and for concerns ν we can write

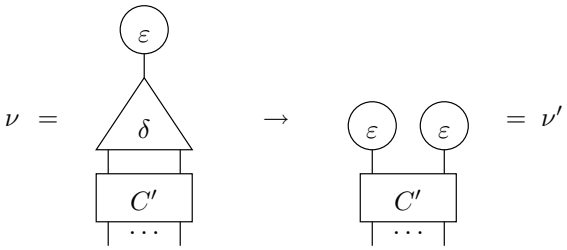


Now it is easy to see that $\mu' \to^* \nu'$, so $\mu' \simeq_\beta \nu'$, and we are done. This establishes property (2); property (4) is proved in a similar way.

We now turn to the $\gamma\varepsilon$ equation. As before, let μ_0 and ν_0 be the resp. the left and right member of the equation, so that $\mu = C[\mu_0]$ and $\nu = C[\nu_0]$ for some context C. Observe that μ_0 and ν_0 are both blind subnets of resp. μ and ν, so they do not contribute to observable paths, which means that $\mu\downarrow$ iff $\nu\downarrow$. Now suppose that $\mu \to \mu'$. Again, if the reduction happens inside C, ν has no problem in simulating. So we can consider the case of a cell interacting with μ_0. If this is a γ cell, the reader can check that $\nu \to \mu'$, so no problem. If it is a δ cell, then we have

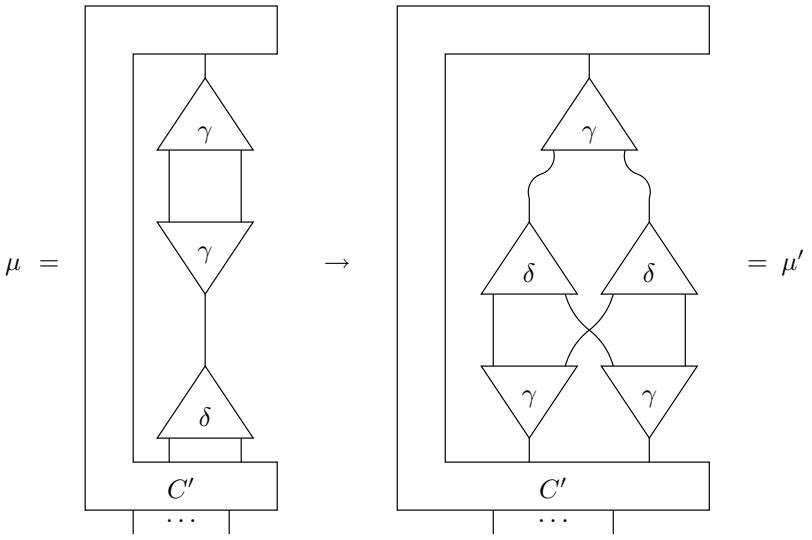


whereas

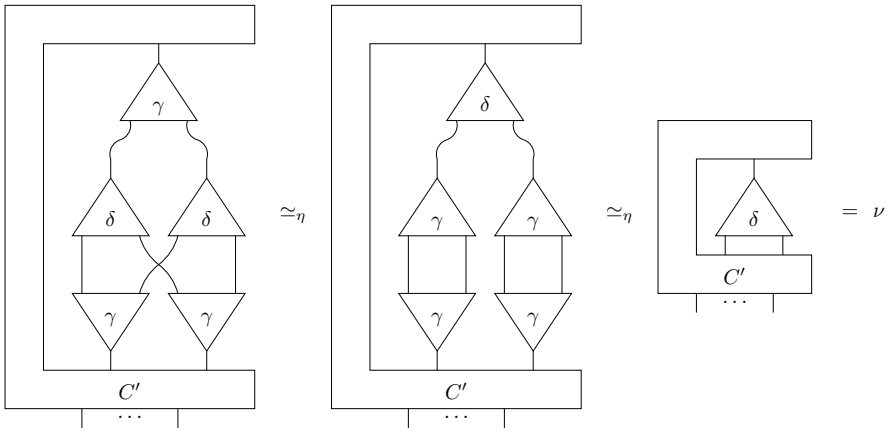


By applying two $\delta\varepsilon$ rules in μ', we obtain a net which is η-equivalent to ν', so $\mu' \simeq_{\beta\eta} \nu'$. The case of an ε cell is trivial. Again, we have established property (2); establishing property (4) is done by applying nearly identical arguments.

The last equation to be taken care of is the $\gamma\gamma$ equation. Again, let μ_0 and ν_0 be resp. the left and right member of the equation (i.e., ν_0 is a wire), and let $\mu = C[\mu_0]$ and $\nu = C[\nu_0]$ for some context C. This equation too preserves observable paths regardless of the direction in which it is applied, so $\mu\downarrow$ iff $\nu\downarrow$ holds as in the previous cases. Now suppose $\mu \to \mu'$. As usual, we consider directly the case in which a cell has interacted with μ_0, otherwise the situation is trivial. This time there are two principal ports in μ_0, but fortunately the net is perfectly symmetrical, so we do not need to (and actually cannot!) distinguish which of the two ports has interacted. The cases in which the cell interacting with μ_0 is a γ or an ε cell are easy: in both cases, the reader can check that μ' is resp. equal and η-equivalent to ν. The case of a δ cell is a little bit more complicated:

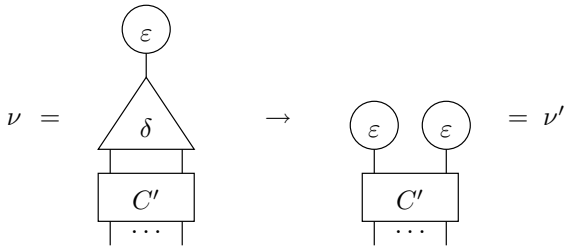


But notice that

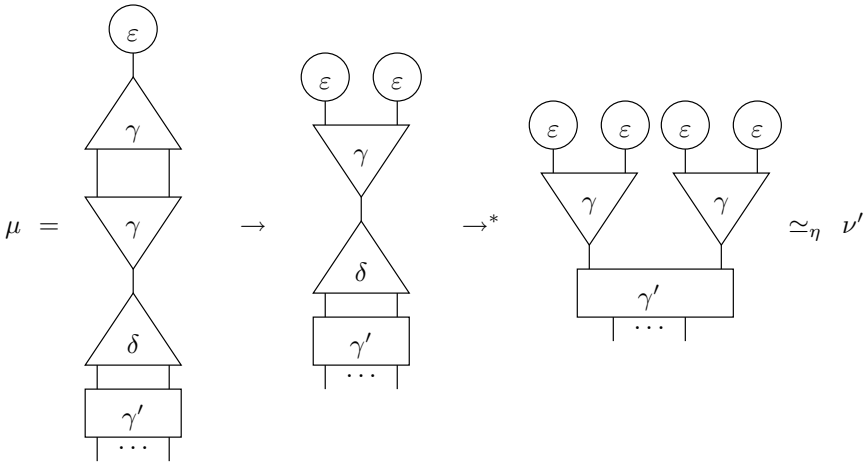


so $\mu' \simeq_\eta \nu$. This is all as far as property (2) is concerned. To prove property (4), suppose that $\nu \to \nu'$. The only interesting situation is that in which the

wire ν_0 is used in the reduction, i.e., ν_0 connects two principal ports in ν. There are six possible cases, one for each active pair. We shall only check the $\delta\varepsilon$ active pair; in all other cases, we let the reader verify that $\mu \to^* \nu'$. The situation is the following:



In μ, we have



so μ is able to reduce to a net η-equivalent to ν', and we are done. □

Proposition 2.15 *Let μ, ν be two nets. Then, $\mu \simeq_{\beta\eta} \nu$ implies $\mu \simeq \nu$.*

Proof. An easy corollary of Lemma 2.14 and of the fact that $\simeq_{\beta\eta}$ is a congruence: suppose $\mu \simeq_{\beta\eta} \nu$, and let C be any suitable context; by the congruence property, we have $C[\mu] \simeq_{\beta\eta} C[\nu]$, and by Lemma 2.14 we obtain $C[\mu] \stackrel{\cdot}{\sim} C[\nu]$. No assumption has been made on C (except that of being an adequate context for μ and ν, which by the way must have the same number of free ports because they are $\beta\eta$-equivalent), so we can conclude that $\mu \simeq \nu$ thanks to Corollary 2.11. □

In Sect. 2.3.3 we shall prove that, if we restrict to total nets, then the converse of Proposition 2.15 also holds; this is the analogue of Bhöm's Theorem in the λ-calculus.

2.3.2 Termination

In this section we give a sufficient condition for a net to be total, formulated in terms of straight paths (Definition 2.1). In the case of a generic INS, interaction paths should be used; however, as already remarked, these coincide with straight paths in the interaction combinators.

Definition 2.17 (Maximal path) *A* maximal path *in a net μ is a straight path ending into a free port of μ or into the principal port of an ε cell of μ.*

In the following, if μ is a net and α a cell of μ, we say that a straight path ϕ *starts from* α if $\phi = px\phi'$, where p is the principal port of α, and x is *not* an auxiliary port of α.

Definition 2.18 (Well-founded net) *A net μ is* well-founded *iff for each cell α of μ, there is a finite non-null number of maximal paths starting from α.*

Basically, the definition above assures that in a well-founded net there are no infinite straight paths.

Lemma 2.16 (Stability of well-foundedness under reduction) *Let μ be a net such that $\mu \to \mu'$. If μ is well-founded, then so is μ'.*

Proof. By simple inspection of the reduction rules. □

Proposition 2.17 *If a net is well-founded, then it is total.*

Proof. Let μ be a well-founded net. First of all observe that if μ contains a vicious circle, then by definition there is a cell of μ (maybe a "virtual cell", in the case of a cyclic wire) admitting no maximal path starting from it. Therefore, the well-foundedness of μ implies the absence of vicious circles in μ, and by Lemma 2.16 also in any reduct of μ.

We need only show then that the reduction of μ terminates. For this, let μ_0 be a generic well-founded net, and let α be a cell of μ_0. We define the *weight* of α, denoted $\sharp(\alpha)$, as the sum of the lengths of all maximal paths starting from α and crossing at least one active pair, i.e., containing a sequence pq where p and q are the principal ports of two cells of μ_0. By definition of well-founded net, this is a non-negative integer. Then, we define the weight of μ_0, still denoted $\sharp(\mu_0)$, as

$$\sharp(\mu_0) = \sum \sharp(\alpha),$$

where the sum is taken over all cells of μ_0; this too is clearly a non-negative integer.

We now prove termination by induction on $\sharp(\mu)$:

- $\sharp(\mu) = 0$. This is equivalent to saying that μ contains no active pair. In fact, the presence of an active pair immediately yields two cells α, α' such that $\sharp(\alpha), \sharp(\alpha') > 0$; for the converse, by definition the absence of active pairs implies $\sharp(\alpha) = 0$ for all α.

- $\sharp(\mu) > 0$. By the previous remark, μ contains an active pair. We reduce it, obtaining μ', and we show that $\sharp(\mu') < \sharp(\mu)$ (remember that, thanks to Lemma 2.16, the weight of μ' is defined). Then we apply the induction hypothesis: since μ' is a reduct of μ, if μ' is total then so is μ.

  Proving that the weight strictly decreases is done by a case-by-case analysis of the six reduction rules. The only interesting case is that of the $\gamma\delta$ rule, since in all other cases the number of cells strictly decreases, and the paths involved in the rules are shortened.

  So let α be one of the two cells involved in a $\gamma\delta$ active pair. The rule being perfectly symmetrical, we can consider only α and one its two copies

after the application of the rule, which we call α_1 and α_2. To make the situation even more symmetrical, it does not harm to assume that both cells involved in the active pair have weight $\sharp(\alpha)$, so that $\sharp(\mu) = m+2\cdot\sharp(\alpha)$ and $\sharp(\mu') = m'+2\cdot\sharp(\alpha_1)+2\cdot\sharp(\alpha_2)$, where m and m' are suitable integers.

First of all, observe that for each straight path ϕ "passing through" the active pair in μ, there is a straight path ϕ' *of the same length* in μ', and vice versa. This means that $m' = m$.

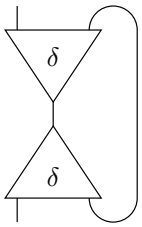
Now, any maximal path starting from α must pass through one of the two auxiliary ports of the cell that α is interacting with. This means that the set of maximal paths starting from α can be partitioned into two. Moreover, in any case a maximal path starting from α has length at least 2. Therefore, $\sharp(\alpha) = \sum(w_i^1 + 2) + \sum(w_i^2 + 2)$, where the w_i^j are suitable non-negative integers, and the sums are taken over the paths in each element of the partition.

If we turn to α_1, we see that all maximal paths starting from it "come from" the maximal paths starting from α of one of the two components of the partition considered above. The same holds for α_2, so we can write, for $j \in \{1, 2\}$, that $\sharp(\alpha_j) = \sum w_i^j$.

Now clearly $\sum w_i^1 + \sum w_i^2 < \sum(w_i^1 + 2) + \sum(w_i^2 + 2)$, so we are done.

□

We observe that well-foundedness is *not* a necessary condition for a net to be total. To see why, it is enough to consider the net

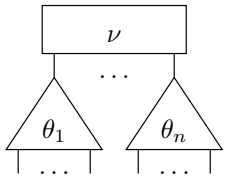


This net is not well-founded, since there is an infinite number of maximal paths starting from both of the cells it contains. And yet, the net is total, because it reduces in one step to a wire. This shows in particular that well-foundedness is *not* stable under anti-reduction.

The reader acquainted to the Geometry of Interaction (GoI) may see that well-foundedness is indeed a stronger version of *nilpotency* [Gir89, Laf97]. In the GoI semantics, fewer straight paths are taken into consideration, namely those that have a non-null weight in the dynamic algebra, or *regular paths* in Danos&Regnier's terminology. It is possible to show that well-foundedness formulated in terms of maximal regular paths, i.e., finiteness of regular paths, becomes also a necessary condition for a net to be total, and thus nilpotency characterizes total nets. This will be proved in Sect. 3.3, Theorem 3.43.

Although weaker, Proposition 2.17 is however enough for our present purposes; in fact, it suffices to prove the following result, which will be constantly (and often silently) used in the rest of the section.

Lemma 2.18 *Let ν be a reduced net with $n > 0$ free ports, and let $\theta_1, \ldots, \theta_n$ be tests. Then, the net*



is total.

Proof. It is not hard to check that the above net is well-founded, therefore total by Proposition 2.17. □

In particular, Lemma 2.18 shows that the net obtained by plugging any two trees by their roots is total.

2.3.3 The separation theorem

We now formulate and prove an internal separation result similar to Böhm's Theorem in the λ-calculus: given two non-$\beta\eta$-equivalent nets, we build a context separating them.

It is important to observe however that "separating" does not have exactly the same sense as in Böhm's classical result: in the λ-calculus, two distinct $\beta\eta$-normal forms can be separated by sending them to *any* pair of distinct terms (the typical choice being the projections $\lambda xy.x$ and $\lambda xy.y$); in the interaction combinators, the uninformative behavior of the ε combinator forces it to be one of the separation values, as no context can extract any information from it. Therefore, we actually obtain something more akin to Hyland's Theorem (sometimes referred to as "semi-separation"), which extends Böhm's result to non-normal terms. This reveals a sharp difference between interaction combinators and the λ-calculus, as "full" separation already fails for normal nets.

Nevertheless, the separation is still strong enough to have important semantical consequences, i.e., the maximality of $\beta\eta$-equivalence: any non-trivial denotational semantics of the interaction combinators must distinguish non-$\beta\eta$-equivalent total nets (cf. Theorem 2.27).

The internal separation result will be formulated for nets with only one free port. This is just a convenient choice; taking into account nets with more than one free port is a straight-forward generalization, as will become clear by looking at the techniques used in the proof.

If μ is a net with one free port and θ a test, we write $\theta[\mu]$ for the net obtained by plugging the only free port of μ into the only free principal port of θ (or into any of its free ports in case θ is a wire). In what follows, we write E for the net with two free ports consisting of two ε combinators, and W for the net consisting of a single wire.

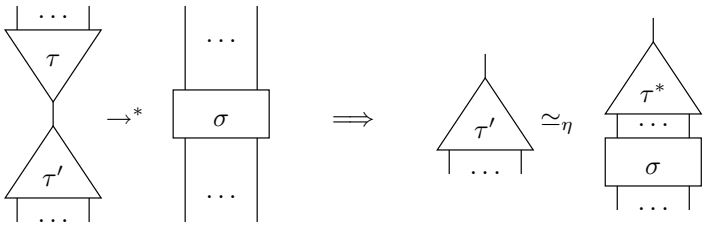
Theorem 2.19 (Separation) *Let μ, μ' be two total nets with one free port such that $\mu \not\approx_{\beta\eta} \mu'$. Then, there exists a 2-test θ such that $\theta[\mu] \to^* E$ and $\theta[\mu'] \to^* W$,* or vice versa.

The Separation Theorem states in particular that, on total nets, $\simeq_{\beta\eta}$ and $\simeq$ coincide. In fact, the values used for separating two non-$\beta\eta$-equivalent nets are the prototypical examples of an observable (W) and a blind (E) net. Additionally, the theorem improves the Context Lemma 2.5 given in Sect. 2.1: not all principal contexts are needed to discriminate nets, but only *tests* suffice, i.e., principal contexts in which there is no connection between the leaves of the trees that compose them.

The rest of the section is devoted to the proof of Theorem 2.19. A few intermediate results are needed, which we go through in the sequel.

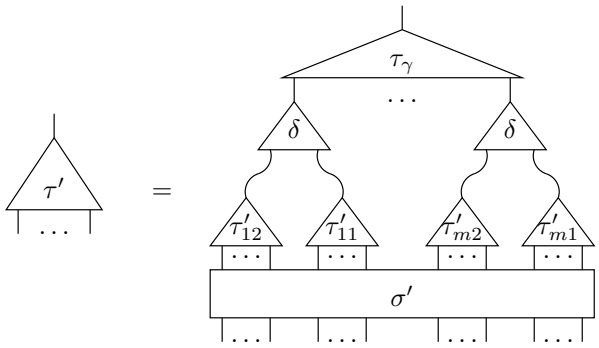
Lemma 2.20 *Anti-trees are unique up to η-equivalence, i.e., if τ' is an anti-tree of τ, then $\tau' \simeq_\eta \tau^*$.*

Proof. We shall prove the following implication:

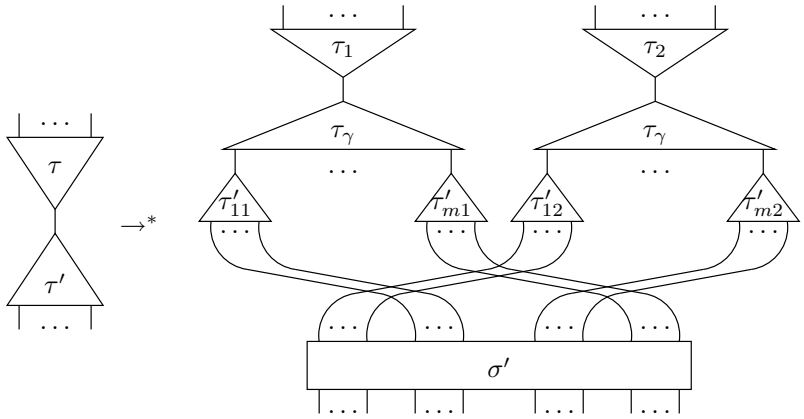


where σ is a generic wiring. The statement of the lemma is obviously a special case of it.

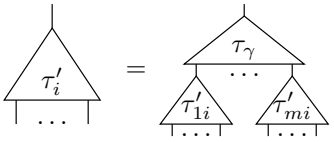
We reason by induction on τ. If $\tau = \mathbf{1}$, the result is obvious. Let then $\tau = \delta(\tau_1, \tau_2)$. The key observation is that



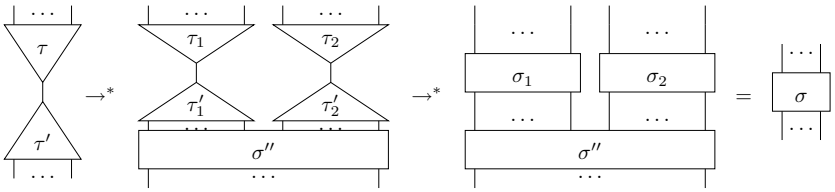
where τ_γ is the maximal subtree of τ' containing no δ cell. In fact, all leaves of τ_γ must be connected to a δ cell; if it were otherwise, a leaf of τ_γ would be free, so that τ and τ' would not reduce to a wiring. Hence we have



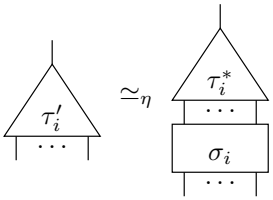
Now, if we put, for $i \in \{1, 2\}$,



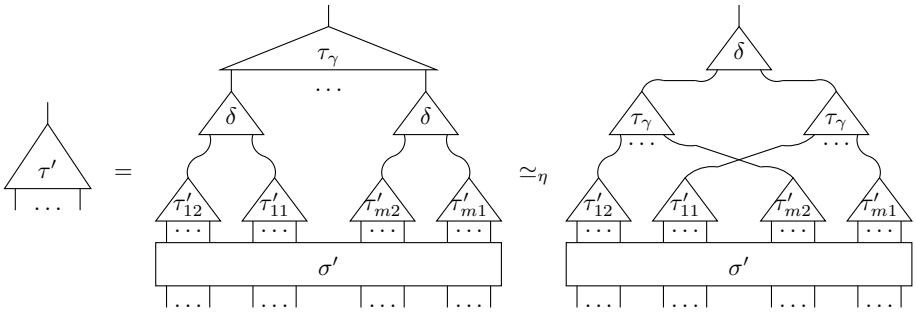
and if we absorb σ' and the other wirings into a wiring called σ'', we obtain



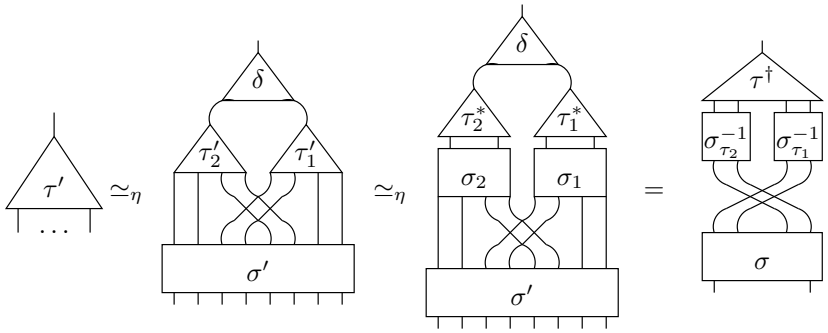
where σ_1 and σ_2 must be wirings because by hypothesis σ is a wiring. Now we can apply the induction hypothesis as follows:



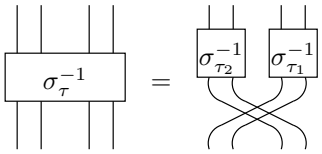
for $i \in \{1, 2\}$. We shall not do it explicitly here, but using the $\gamma\delta$ equation, it is possible to prove by induction on τ_γ that



from which we obtain

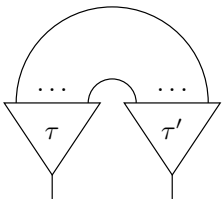


Now the reader can check that, since $\tau = \delta(\tau_1, \tau_2)$, we have



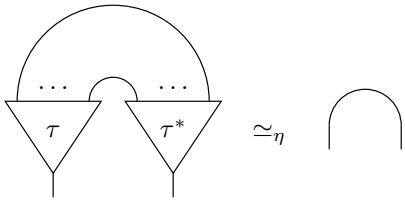
which proves what we wanted (see the definition of canonical anti-tree, p. 74). A similar argument can be given for the case $\tau = \gamma(\tau_1, \tau_2)$; the details are left to the reader. □

Lemma 2.21 (Wire characterization) *Let ν be a cut-free net with two free ports. Then, ν is η-equivalent to a wire iff it has the following shape:*



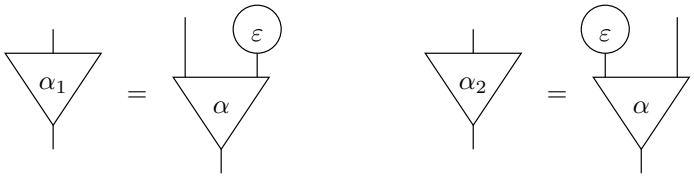
where τ' is an anti-tree of τ.

Proof. The "only if" part is proved by induction on the number of η-equations applied to "expand" the wire. For what concerns the "if" part, by Lemma 2.20 we have $\tau' \simeq_\eta \tau^*$. It is then not hard to prove that

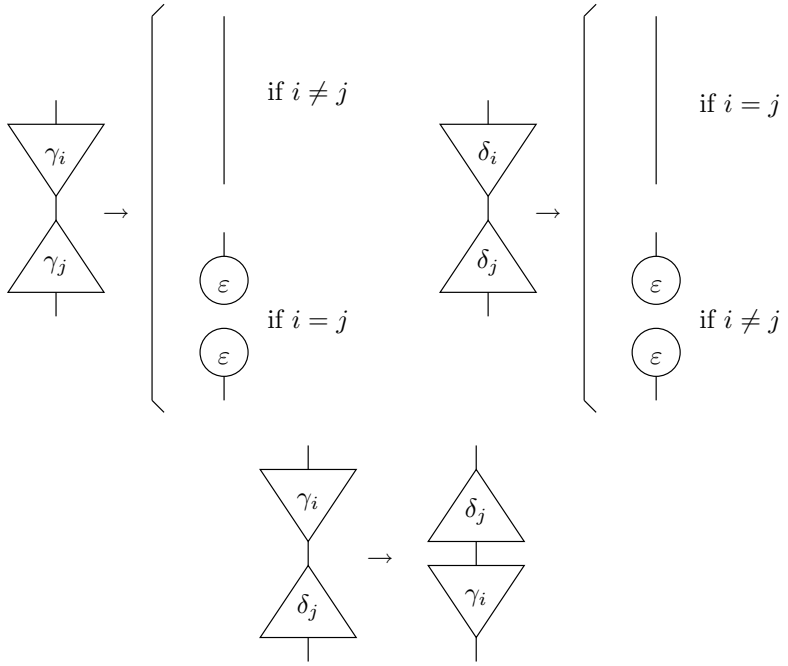


by induction on τ. The details are left to the reader. □

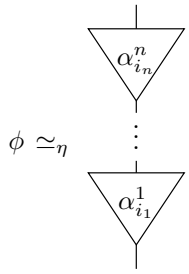
Consider now the principal nets $\gamma_1, \gamma_2, \delta_1, \delta_2$ defined as follows:



where $\alpha \in \{\gamma, \delta\}$. These nets can be seen as cells behaving according to the following interaction rules:

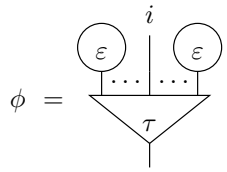


Now, if ϕ is a filiform tree, using the two equations of η-equivalence concerning the ε combinator, one can see that, for some n, we have



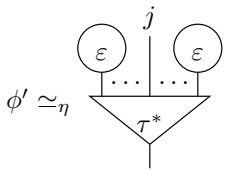
where $\alpha^j \in \{\gamma, \delta\}$ and $i_j \in \{1, 2\}$. For this reason, we shall identify filiform trees with finite words over the alphabet $\{\gamma_1, \gamma_2, \delta_1, \delta_2\}$; the example above corresponds to the word $\alpha^1_{i_1} \cdots \alpha^n_{i_n}$. We can thus show the following:

Lemma 2.22 *Let ϕ be a filiform tree such that*



where we have numbered the leaves of τ from 1 to n and we have supposed that the only one not connected to an ε cell is the ith one. Let now ϕ' be another filiform tree, and call μ the net obtained by plugging ϕ and ϕ' together by means of their roots. Then

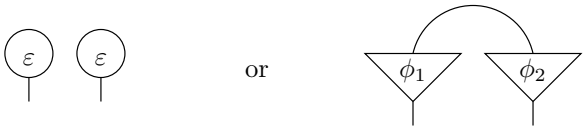
1. *μ reduces to a wire iff*



   where $j = n - i + 1$;

2. *if μ does not reduce to a wire, then its cut-free form contains ε cells.*

PROOF. We start by proving part (ii). Considering that any filiform tree is η-equivalent to a sequence of cells interacting as described above, the reader will easily convince him/herself that the cut-free form of μ must be η-equivalent to one of the following nets:



where ϕ_1, ϕ_2 are themselves filiform trees. In any case, if one of ϕ_1, ϕ_2 is different from $\mathbf{1}$ (i.e., the cut-free form of μ is not a single wire), then the cut-free form of μ contains ε cells, because the presence/absence of ε cells is preserved under η-equivalence.

Let us turn to part (i). The "if" direction is a direct consequence of the property defining anti-trees (see p. 74). For the converse, suppose that μ reduces to a wire. By the above remark, we know that $\phi \simeq_\eta \phi_0 = \alpha_1 \cdots \alpha_m$ and $\phi' \simeq_\eta \phi'_0 = \alpha'_1 \cdots \alpha'_{m'}$, where the α_k and α'_k are elements of $\{\gamma_1, \gamma_2, \delta_1, \delta_2\}$. Call μ_0 the net obtained by plugging ϕ_0 and ϕ'_0 together by means of their roots. Clearly $\mu_0 \simeq_\eta \mu$, so μ_0 is $\beta\eta$-equivalent to a wire. Now let ω_0 be the cut-free form of μ_0. It is enough to inspect the interaction rules introduced at p. 87 to see that the "filiform structure" is preserved under reduction, i.e., no reduct of μ_0 has more than two auxiliary ports connected together by a wire. But then, by Lemma 2.21, ω_0 must be a single wire as well.

Now, since cells annihilate in pairs, if μ_0 reduces to a wire we must have $m' = m$; additionally, for each $\alpha_k = \gamma_1$ (resp. $\alpha_k = \gamma_2$), there must be exactly one $\alpha'_l = \gamma_2$ (resp. $\alpha'_l = \gamma_1$), and for each $\alpha_k = \delta_1$ (resp. $\alpha_k = \delta_2$), there must be exactly one $\alpha'_l = \delta_1$ (resp. $\alpha'_l = \delta_2$), and no γ_1 must "meet" a γ_1, or a γ_2 "meet"

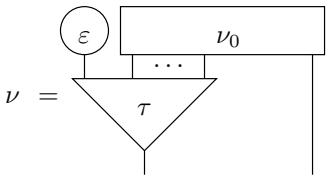
a γ_2, etc. In other words, we must have $\phi'_0 \simeq_\eta \alpha_1^* \cdots \alpha_m^*$, where we put $\gamma_1^* = \gamma_2$, $\gamma_2^* = \gamma_1$, $\delta_1^* = \delta_1$, and $\delta_2^* = \delta_2$. It is not hard to see that this corresponds to the canonical anti-tree of τ. □

In the following, if ν is a net with two free ports, we use the notation $(\phi_1, \phi_2)[\nu]$ to denote the net obtained by plugging the roots of two filiform trees ϕ_1, ϕ_2 into the free ports of ν.

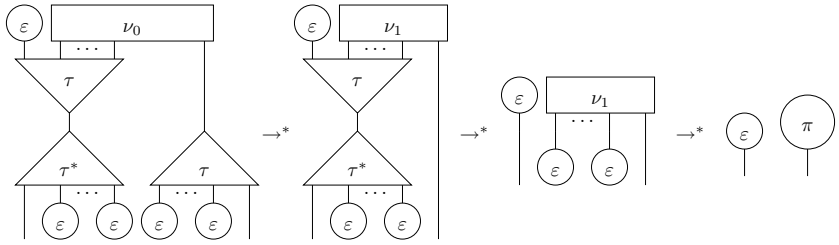
Lemma 2.23 (Wire separation) *Let ν be a cut-free net with two free ports, such that $\nu \not\simeq_\eta W$. Then, there exist two filiform trees ϕ_1, ϕ_2 such that $(\phi_1, \phi_2)[W] \to^* W$ and $(\phi_1, \phi_2)[\nu] \to^* E$, or $(\phi_1, \phi_2)[W] \to^* E$ and $(\phi_1, \phi_2)[\nu] \to^* W$.*

Proof. Since $\nu \not\simeq_\eta W$, by Lemma 2.21 we have three possibilities:

- ν contains at least one ε cell. In this case, we will show how to send ν to E, and W to itself. First of all, we can assume w.l.o.g. that ν has the following shape:

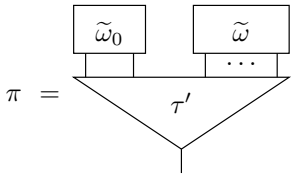


  where ν_0 is a reduced net. Now, consider an anti-tree τ^* with ε cells plugged on every leaf except the one corresponding to the ε cell present in ν. What we obtain is a filiform tree, and the same happens if we repeat the construction on τ. We then have

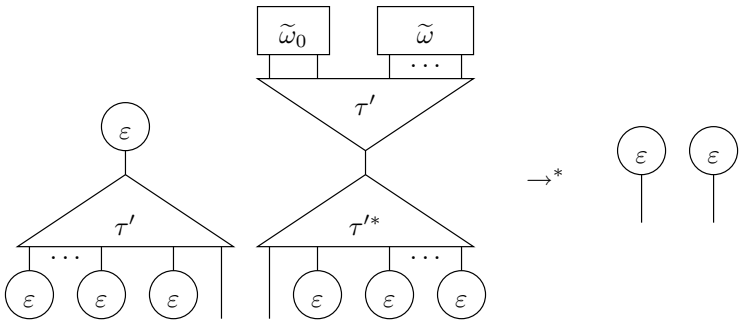


  for a suitable reduced net ν_1 and package π. Notice that, on the other hand, when the same two filiform trees are plugged to the two extremities of the wire ν', they annihilate and we obtain again a wire.

  If $\pi = \varepsilon$, we are done; otherwise, w.l.o.g. π can be assumed to be of the form

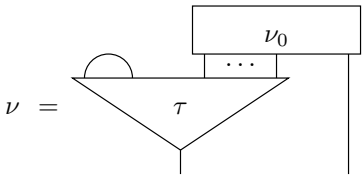


where $\widetilde{\omega}_0$ is W if π is not an ε-package, or E otherwise. Hence, we can consider two more filiform trees as in the following net, the reduction of which gives

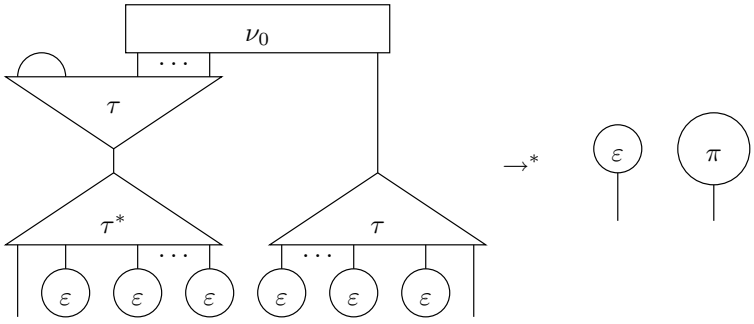


regardless of the nature of $\widetilde{\omega}_0$. When plugged together through a wire, these two new filiform trees yield once again a wire. We have thus realized our goal by using a context consisting of two trees which are in turn compositions of two filiform trees, and are therefore themselves filiform.

- ν contains no ε cell, and there is a maximal path starting from one of the free ports of ν and ending into the same free port; in this case too we show how to send ν to E and W to itself. W.l.o.g., we can assume

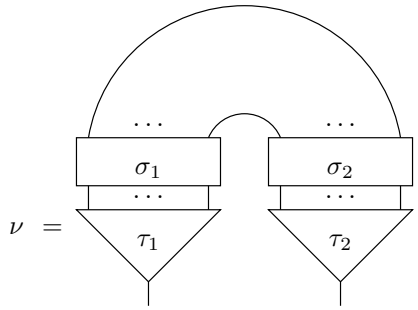


for a suitable reduced net ν_0. Then, we consider two filiform trees which yield (for some package π) the following reduction:

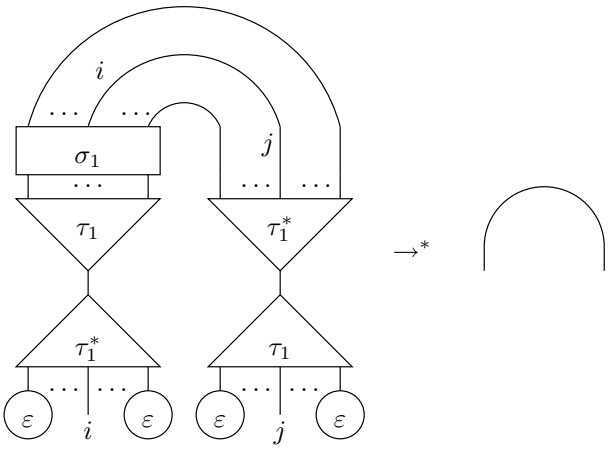


Now we are back to a situation already met in the first case, which we know how to handle.

- ν contains no ε cell, and all maximal paths starting from a free port lead to the other. This time we show how to send ν to W and W to E. The situation is the following:

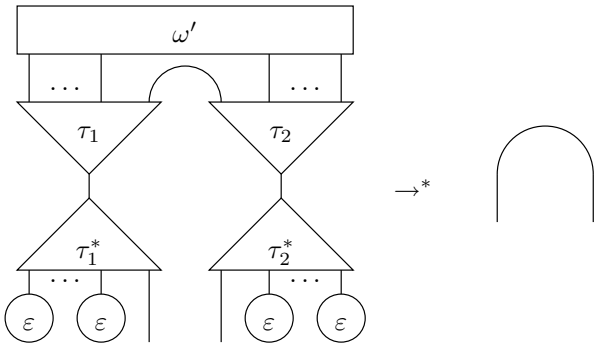


Suppose first that τ_2 and σ_2 form an anti-tree τ_1' of τ_1; by Lemma 2.20, it does not harm to suppose that $\tau_1' = \tau_1^*$. Then, by Lemma 2.21 and by our assumption that $\nu \not\simeq_\eta W$, σ_1 must contain a crossing of wires, so that there exists a leaf of τ_1 and a leaf of τ_1^* which are not "symmetrical" but are connected by a wire. More precisely, if we number the leaves of τ_1 and τ_1^* from 1 to n, there is a connection between a leaf i of τ_1 and a leaf j of τ_1^* such that $j \neq n - i + 1$. Then, we can extract this connection using two filiform trees as follows:



By Lemma 2.22, when the same two filiform trees interact with each other through W, they yield a net containing ε cells. But then we can stop here, since we are back to our first case.

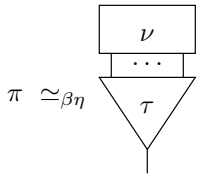
In case τ_2 and σ_2 do not form an antitree of τ_1 (and τ_1 and σ_1 do not form an anti-tree of τ_2), the situation is simpler; we can assume w.l.o.g. that there is a wire linking the "rightmost" leaf of τ_1 to the "leftmost" leaf of τ_2, which can be extracted using two filiform trees as follows:



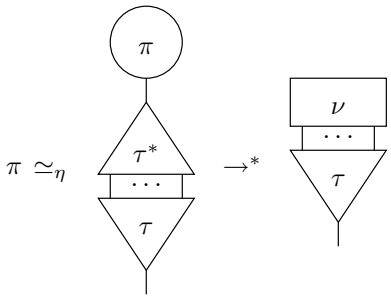
Now by Lemma 2.20 τ_1^* and τ_2^* are certainly not anti-trees of each other, so by Lemma 2.22 their interaction through W produces a net which always contains at least an ε cell, just as above.

□

Lemma 2.24 (Equivalence lemma) *Let π be a package. Then, for any tree τ with n leaves, there exists a cut-free net ν with n free ports such that*



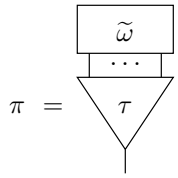
Proof. Applying in the order Lemmas 2.21 and 2.18, we have



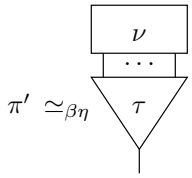
for a suitable cut-free net ν. □

We can at last go into the proof of the Separation Theorem 2.19. First of all, if μ is a total net with one free port and π its cut-free form, by confluence we have that for any test θ, $\theta[\mu]$ and $\theta[\pi]$ have the same cut-free form, therefore it is enough to prove our result for packages.

So let π, π' be two packages such that $\pi \not\simeq_\eta \pi'$. Suppose that



By the Equivalence Lemma 2.24, there exists a cut-free net ν such that



so it does not harm to assume that π and π' "end" with the same tree.

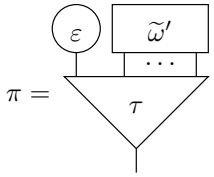
Now, at least one of the following two situations must apply:

(a) there exists a leaf of τ which is connected to an ε combinator of $\widetilde{\omega}$, but is connected to something not η-equivalent to ε in ν;

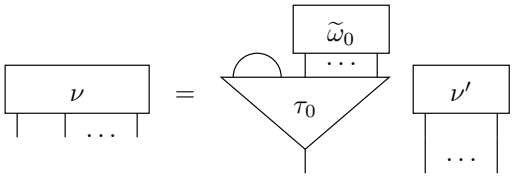
(b) there exist two leaves of τ which are connected by a wire of $\widetilde{\omega}$, whereas in ν the same two leaves are either not connected, or their connection is not η-equivalent to a wire.

As a matter of fact, if neither (a) nor (b) applied, we would have proved that $\pi \simeq_\eta \pi'$, against our hypothesis.

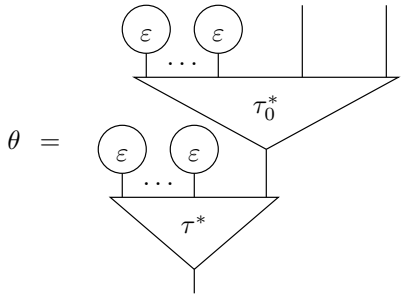
Suppose that situation (a) applies, and suppose w.l.o.g. that the leaf in question is the "leftmost" one, i.e., we have



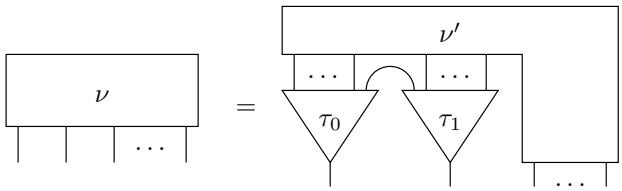
By hypothesis, the "leftmost" free port of ν, let us call it x, is connected to something not η-equivalent to ε; this means that if we "go up" the tree rooted at x in ν, let us call it τ_0, we must find a leaf of τ_0 connected by a wire to some other tree of ν. It may happen that all connections are within τ_0 itself, i.e., we have



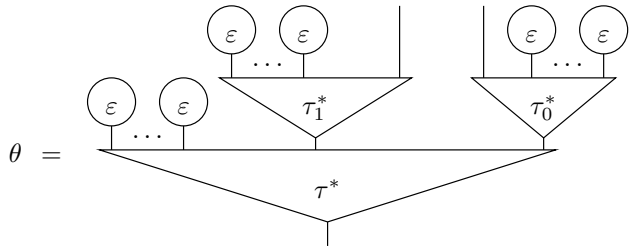
(for graphical convenience, we have assumed w.l.o.g. that there is a direct connection between the "leftmost" two leaves of τ_0). Under such assumptions, one can verify that the 2-test



is such that $\theta[\pi'] \to^* W$, whereas $\theta[\pi] \to^* E$. Suppose instead that τ_0 is connected to some other tree of ν, and suppose w.l.o.g. that this tree is the one immediately "to the right" of τ_0, let us call it τ_1:

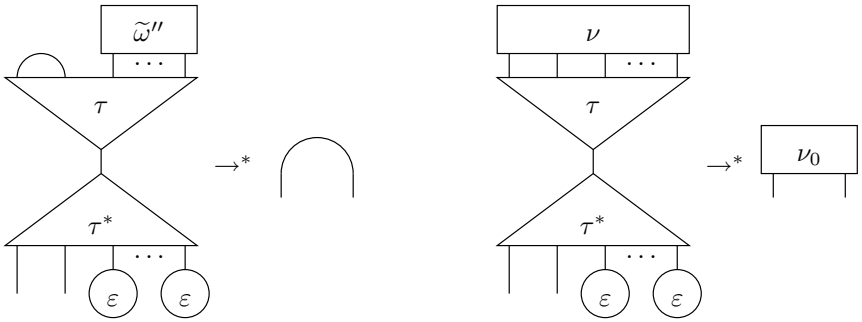


(again, in the picture we have made a convenient assumption about the connection between τ_0 and τ_1, without affecting the generality of our argument). In this case, one may check that the 2-test



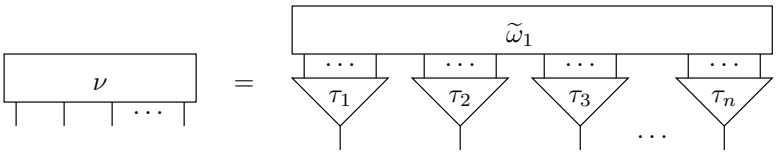
is such that $\theta[\pi'] \to^* W$ and $\theta[\pi] \to^* E$.

Let us now consider situation (b), i.e., π has a direct connection for τ which π' is missing. Then, we can use an anti-tree τ^* and isolate the two leaves involved in the connection:

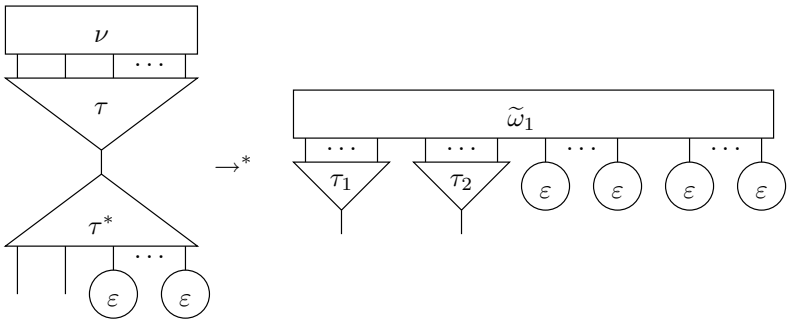


(as usual, for graphical purposes we have supposed w.l.o.g. that the two leaves in question are the "leftmost" ones). It is not too hard to show that, under the hypotheses we have, ν_0 cannot be η-equivalent to a wire. In fact, there are two cases, depending on the shape of ν. The trivial case is when

in which $\nu_0 \not\simeq_\eta W$ by hypothesis. The other case is that in which the two trees "above" the two "leftmost" free ports of ν are connected to the rest of the net, i.e., we have

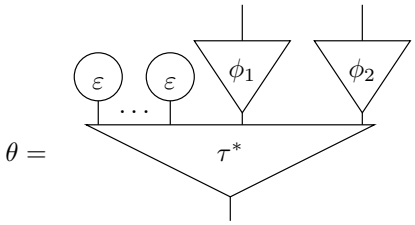


and in $\widetilde{\omega}_1$ there at least one wire connecting a leaf of τ_1 or τ_2 to a leaf of one of the τ_i, for $i \geq 3$. In this case, we have



in which, thanks to the supposed connection, we see that there is at least one leaf of τ_1 or τ_2 connected to an ε cell. But this means that ν_0 contains at least one ε cell, which by Lemma 2.21 entails $\nu_0 \not\simeq_\eta W$.

Hence, the Wire Separation Lemma 2.23 applies, giving us two filiform trees ϕ_1 and ϕ_2 which are able to distinguish between the wire and ν_0. Therefore, if we define



we have $\theta[\pi] \rightarrow^* E$ and $\theta[\pi'] \rightarrow^* W$, or viceversa, which completes the proof.

Notice that the canonical anti-tree of τ contains γ or δ cells only if τ does; since the filiform trees of Lemma 2.23 are also built out of canonical anti-trees, we get the following for free:

Corollary 2.25 (Internal separation for fragments) *Theorem 2.19 holds also for the fragments $\gamma\varepsilon$ and $\delta\varepsilon$ of the interaction combinators.*

We conclude by showing how the Separation Theorem can be easily generalized to nets with arbitrary (non-empty) interfaces, as stated at the beginning of the section. We first need the following lemma:

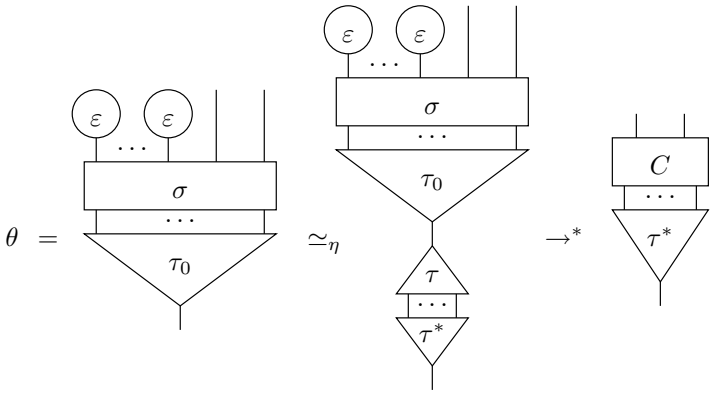
Lemma 2.26 *Let μ, ν be two nets defined as follows:*

where τ is a tree and μ_0, ν_0 two generic nets with at least as many free ports as the number of leaves of τ. Then, $\mu \simeq_{\beta\eta} \nu$ implies $\mu_0 \simeq_{\beta\eta} \nu_0$.

PROOF. The result is obvious: since τ is equal in both nets, and since it can never be involved in any active pair, all the rewriting work applied to show that μ and ν are $\beta\eta$-equivalent must take place inside μ_0 and ν_0. □

Let now μ, ν be two total nets with $n > 1$ free ports such that $\mu \not\simeq_{\beta\eta} \nu$. We choose an arbitrary tree τ with n leaves, an form two nets with one free port μ' and ν' as follows:

By Lemma 2.26, we still have $\mu' \not\simeq_{\beta\eta} \nu'$. Therefore, Theorem 2.19 applies, giving us a 2-test θ such that, for example, $\theta[\mu'] \to^* E$ and $\theta[\nu'] \to^* W$. But, applying Lemma 2.21 and Lemma 2.18 as in the proof of the Equivalence Lemma 2.24, we get



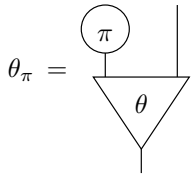
where C is a suitable cut-free net. From this we deduce $C[\mu] \simeq_{\beta\eta} E$ and $C[\nu] \simeq_{\beta\eta} W$, so we have found a context separating μ and ν.

2.3.4 Maximality

As anticipated above, internal separation implies the maximality of $\beta\eta$-equivalence (and thus of $\simeq$) on total nets:

Theorem 2.27 (Maximality) *$\simeq_{\beta\eta}$ is the greatest non-trivial congruence on total nets containing $\simeq_\beta$, i.e., if $\sim$ is a congruence on total nets such that $\simeq_\beta \subseteq \sim$, then either $\sim \subseteq \simeq_{\beta\eta}$, or $\mu \sim \mu'$ for all total nets μ, μ'.*

PROOF. Once again, we prove the result in case μ and μ' have only one free port; the general case follows easily. If $\sim$ is a congruence such that $\simeq_\beta \subseteq \sim$, and if $\mu \sim \mu'$ for two total nets with one free port such that $\mu \not\simeq_{\beta\eta} \mu'$, by Theorem 2.19 we have a 2-test θ such that, for example, $\theta[\mu] \to^* E$ and $\theta[\mu'] \to^* W$. Now put

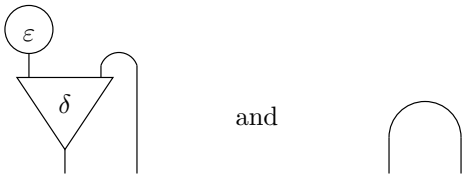


where π is any package. It is not hard to verify that plugging μ into the free principal port of θ_π yields a net which reduces to ε (i.e., the ε combinator), while doing the same with μ' yields a net reducing to π. But $\sim$ is a congruence and is preserved through reduction, so $\mu \sim \mu'$ implies $\varepsilon \sim \pi$, for all π. By transitivity of $\sim$, and by its stability under reduction, we conclude that it must identify all total nets with one free port. □

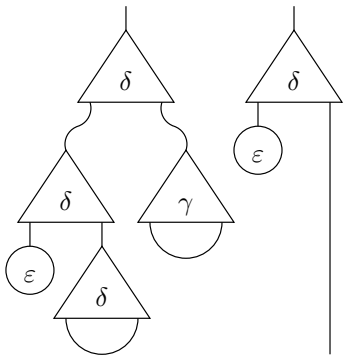
2.3.5 On the strength of separation

We have already remarked that the separation achieved by Theorem 2.19 is in some sense "weaker" than that of Böhm's Theorem, because of its asymmetry: the *or vice versa* in the statement of the theorem is necessary, and cannot be controlled, i.e., there are pairs of nets that "force" a certain separation, refusing the symmetrical one (think of the ε package paired with any other non-$\beta\eta$-equivalent package).

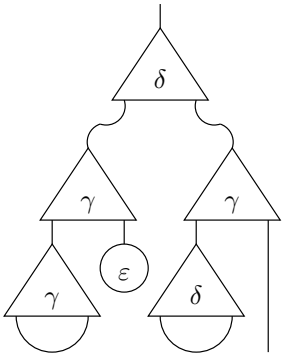
In spite of this, there are nets that can be separated in a "stronger" way. For example, take the two nets



If we call γ (resp. δ) the package consisting of a single γ (resp. δ) cell with its auxiliary ports connected by a wire, we invite the reader to check that the context

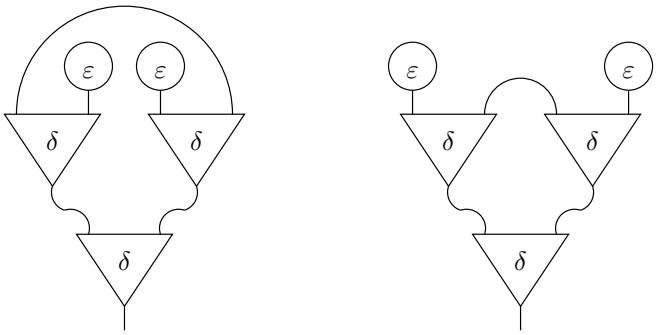


sends one of the above nets to γ, and the other to δ. The reader can also check that the net



"exchanges" γ and δ; therefore, this kind of separation is symmetrical, just as in Böhm's Theorem.

There are also "intermediate" situations, in which, although Theorem 2.19 holds with an *and vice versa*, there is no principal net of arity 1 achieving "strong" separation, i.e., sending one net to γ and the other to δ. An example is given by the following pair of packages:

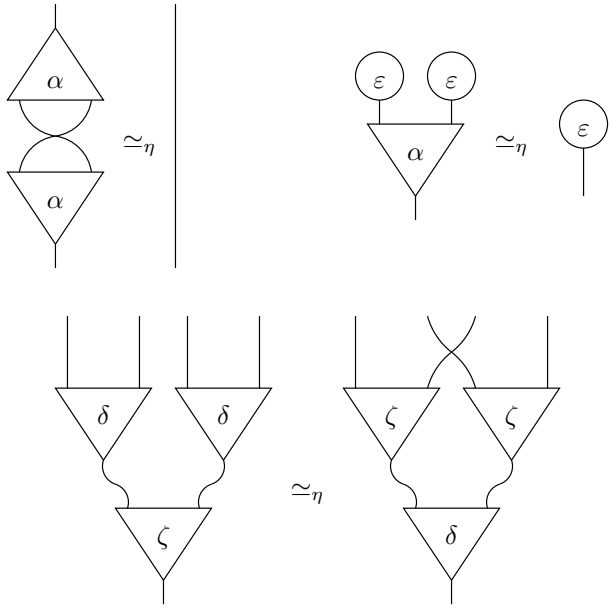


2.3.6 The separation theorem in the symmetric combinators

In this section, we consider the INS of the symmetric interaction combinators, with the obvious conventions on the terms "cell", "net", and "rule".

A notion of η-equivalence can be defined also in the symmetric interaction combinators, just as one would expect:

Definition 2.19 (η-equivalence, symmetric combinators) *Two nets μ, μ' are* η-equivalent*, notation $\mu \simeq_\eta \mu'$, iff they can be rewritten one into the other by means of the following equalities (applied any number of times and under any context):*



where $\alpha \in \{\delta, \zeta\}$.

As usual, $\beta\eta$-equivalence is defined as the transitive closure of $\simeq_\beta \cup \simeq_\eta$. Since the structure of the rules for the symmetric combinators is nearly identical to that of the interaction combinators, there is no surprise in checking that all of the results needed to prove internal separation hold without any problem. Therefore, we have

Theorem 2.28 (Separation, symmetric combinators) *The Separation Theorem 2.19 holds also for the symmetric combinators.*

As we shall see, internal separation in the symmetric combinators will be of crucial importance in the semantical study of Sect. 3.1.

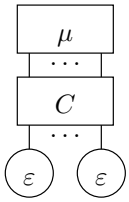
2.4 Totality-based observational equivalence

The observational equivalence discussed so far is based on observing the presence or absence of certain interaction paths upon interaction with a context. Instead of this, we can imagine to observe simply normalization, or rather, totality.

2.4.1 Observing totality

About totality upon immersion into a context, in the case of εINS's we can prove the following:

Lemma 2.29 *If* μ *is a net and* C *an adequate context,* $C[\mu]$ *is total iff the net*



reduces to the empty net.

PROOF. If $C[\mu]$ is total, by definition it reduces to a cut-free net ν, which can be easily proved to be "eaten up" by ε cells. Conversely, if $C[\mu]$ is not total, then it either reduces to a net containing vicious circles, which by their nature can never be eliminated, or each of its reducts contains an active pair, which too cannot be deleted by ε cells. □

Therefore, if we are dealing with an εINS, as it is always the case, we can take as contexts for nets with n free ports... nets with n free ports themselves, which is particularly elegant, because it makes nets and contexts live even more in "the same world" than before. So, in the rest of the section we shall only consider εINS's.

If μ, ν are two nets with the same number of free ports, numbered from 1 to n, we can build a net o with empty an interface by plugging the free port i of μ to the free port $n-i+1$ of ν. This particular way of connecting the two nets is chosen only for convenience: it assures the planarity of our drawings, because in our pictures we always assume that the numbering increases from left to right. Of course any other way of connecting the two interfaces is equivalent, as long as one is fixed once and for all. Now, o is either total (in which case the only possibility is that it reduces to the empty net), or not. In the first case, we write $\ll \mu \mid \nu \gg = \mho$, in the second $\ll \mu \mid \nu \gg = \Omega$.

We can then give the following alternative definition of observational equivalence:

Definition 2.20 *Let μ, μ' be two nets with n free ports. We write $\mu \simeq^\circ \mu'$ iff, for any net with n free ports ν, $\ll \mu \mid \nu \gg = \ll \mu' \mid \nu \gg$.*

Of course we can restrict our quantification over cut-free nets only, since using a non-total net as a context does not give any discriminative power:

Lemma 2.30 (Context) *If $\mu \not\simeq^\circ \mu'$, then there exists a cut-free net ν such that $\ll \mu \mid \nu \gg \neq \ll \mu' \mid \nu \gg$.*

The difference between $\simeq$ and $\simeq^\circ$ is in spirit akin to the difference between observing head normalization and normalization *tout court* in the λ-calculus. For example, a non-total net "producing" observable paths can be equivalent to a cut-free net with respect to $\simeq$ (an example will be given for the interaction combinators, p. 133), while no such equivalence is ever possible with respect to $\simeq^\circ$. Also, $\simeq^\circ$ is based on an *a priori* distinction between the empty net and all other nets with no free ports, while all nets with an empty interface are identified by $\simeq$.

Nevertheless, contrarily to what happens in the λ-calculus, there is no inclusion between the two equivalences (in the λ-calculus, normalization-based equivalence is strictly more discriminative than the one based on head normalization; in fact, it is well known that the equational theory corresponding to this latter is the greatest sensible λ-theory). As a matter of fact, a non-total net producing observable paths and a blind net are distinguished by $\simeq$, but are identified by $\simeq^\circ$.

2.4.2 Totality-based equivalence in the interaction combinators

In spite of the discussion at the end of the previous section, it is possible to prove that, for total nets of interaction combinators (symmetric or not), the equivalences $\simeq$ and $\simeq^\circ$ coincide.

It is immediate that $\simeq^\circ \subseteq \simeq_{\beta\eta}$. To see this, suppose $\mu \not\simeq_{\beta\eta} \mu'$. By Theorem 2.19, we have a test θ such that, for example, $\theta[\mu] \to^* E$ and $\theta[\mu'] \to^* W$. Then, by connecting the two free auxiliary ports of θ with a wire, we clearly obtain a package π such that $\ll \mu \mid \pi \gg = \mho$ while $\ll \mu' \mid \pi \gg = \Omega$. Moreover, by a similar argument to that given for Proposition 2.15, we can prove that, on total nets, $\simeq_{\beta\eta} \subseteq \simeq^\circ$. Therefore $\simeq$, $\simeq^\circ$, and $\simeq_{\beta\eta}$ coincide on total nets.

The Separation Theorem also shows that, in the case of the combinators, not all cut-free nets are necessary as contexts (as Lemma 2.30 says), but "bundles" of packages already suffice to discriminate between nets. If a net μ has its free ports numbered from 1 to n, we can take n packages $\pi_1, \ldots, \pi_n$ and form a net with an empty interface by plugging each π_i into the free port i of μ; then, we write $\ll \mu \mid \pi_1, \ldots, \pi_n \gg = \mho$ if such a net is total, $\ll \mu \mid \pi_1, \ldots, \pi_n \gg = \Omega$ otherwise.

Then, the Separation Theorem says that whenever μ, μ' are two nets with n free ports such that $\mu \not\simeq_{\beta\eta} \mu'$, there exist n packages $\pi_1, \ldots, \pi_n$ such that $\ll \mu \mid \pi_1, \ldots, \pi_n \gg \neq \ll \mu' \mid \pi_1, \ldots, \pi_n \gg$. This is reminiscent of the Separation Theorem holding for the designs of Girard's *ludics* [Gir01]. In fact, in the following section we shall relate internal separation to topological separation exactly as done by Girard in ludics.

2.4.3 Internal separation and topological separation

Following Girard [Gir01], we can give a topological interpretation to the Separation Theorem 2.19. Call Π the quotient of the set of packages of combinators under $\simeq_{\beta\eta}$. For, $\pi, \pi' \in \Pi$, define $\pi \preceq \pi'$ iff, for all $\rho \in \Pi$, $\ll \pi \mid \rho \gg = \mho$ implies $\ll \pi' \mid \rho \gg = \mho$. The set Π can be endowed with the Alexandrov topology associated to $\preceq$: a set $O \subseteq \Pi$ is Alexandrov open iff it is upper-closed, i.e., if $\pi \in O$ and $\pi \preceq \pi'$, then $\pi' \in O$.

The Separation Theorem states that the Alexandrov topology on Π is T_0. To see why, consider the following. Given $X \subseteq \Pi$, define

$$\sim X = \{\pi' \in \Pi \mid \forall \pi \in X, \ll \pi \mid \pi' \gg = \mho\} .$$

It is not hard to convince oneself that all sets of the form $\sim X$ are Alexandrov open. Now take any two distinct π, π' in Π. This means that $\pi \not\simeq_{\beta\eta} \pi'$; Theorem 2.19 then gives us a package ρ such that $\sim\{\rho\}$ is a neighborhood of π not containing π', or vice versa.

In some cases, the *or vice versa* can be replaced by an *and vice versa*, (cf. Sect. 2.3.5), which means that there exist pairs of packages which are T_1-separable. Nevertheless, there is no hope of achieving T_2 (Hausdorff) separation for the Alexandrov topology: for all π, $\pi \preceq \varepsilon$, so the package ε belongs to all open sets, and the intersection of two neighborhoods can never be empty. It is interesting to remark in this respect the similarity between the ε combinator and the *daimon* of ludics.

Chapter 3

The Symmetric Combinators

3.1 Denotational semantics

As already recalled, interaction nets are a generalization of multiplicative proof-nets. Thus, in seeking a denotational semantics for the symmetric combinators, it seems natural to draw inspiration from linear logic.

The simplest denotational semantics of linear logic is the *relational semantics*[1], which in the multiplicative case is obtained from coherent spaces by simply ignoring the coherence relation: a formula A is interpreted by a set $|A|$, and a proof of A by a subset of $|A|$; no particular structure is attached or required on $|A|$. In categorical terms, the denotational interpretation takes place in the category **Rel** of sets and relations.

The denotational interpretation of a sequent calculus proof is, as usual, defined by induction. More interestingly, the interpretation can also be defined directly on proof-structures (so, in particular, on proof-nets) by means of *experiments* [Gir87a]. This will be our main source of inspiration, as a sequent calculus is obviously not available in our framework.

3.1.1 Companion bijections

The relational semantics of linear logic is typed; in particular, the two multiplicative connectives (which are the only ones of interest to us) are interpreted by the *Cartesian product*: if the formulas A and B are interpreted resp. by $|A|$ and $|B|$, then $A \otimes B$ and $A \parr B$ are interpreted by $|A| \times |B|$. Our nets are not typed, which means that if we see our binary combinators as multiplicative rules, the natural thing to do would be to consider a set $\mathcal{D}$ in bijection with $\mathcal{D} \times \mathcal{D}$, i.e., any infinite set.

The presence of *two* combinators actually requires two bijections, and the $\delta\zeta$ commutation inspires the following:

[1] To our knowledge, relational semantics has not been formally introduced in any particular work. The best way to see it is perhaps as "coherent spaces without coherence"; it has been considered by many as the starting point for building other denotational semantics of linear logic (semantics = relational semantics + structure), as for example in the work of Thomas Ehrhard [Ehr05].

Definition 3.1 (Companion bijections) *Let $\mathcal{D}$ be an infinite set and*

$$\langle\cdot,\cdot\rangle, [\cdot,\cdot] : \mathcal{D}\times\mathcal{D}\to\mathcal{D}$$

two bijections. $\langle\cdot,\cdot\rangle$ and $[\cdot,\cdot]$ are said to be companions *iff, for all $a,b,c,d\in\mathcal{D}$,*

$$\langle[a,b],[c,d]\rangle = [\langle a,c\rangle,\langle b,d\rangle].$$

Companion bijections do exist, as proved by the following example. Let $\mathcal{A}$ be any infinite set, and let $\beta:\mathcal{A}\times\mathcal{A}\to\mathcal{A}$ be a bijection. We denote by $\mathcal{S}(\mathcal{A})$ the set of all infinite sequences of elements of $\mathcal{A}$. Then we define two functions from $\mathcal{S}(\mathcal{A})\times\mathcal{S}(\mathcal{A})$ to $\mathcal{S}(\mathcal{A})$ as follows:

$$\langle d,e\rangle_n = \begin{cases} d_k & \text{if } n=2k \\ e_k & \text{if } n=2k+1 \end{cases}$$

$$[d,e]_n = \beta(d_n,e_n)$$

In other words, $\langle\cdot,\cdot\rangle$ takes two sequences and builds a new one by interleaving them, while $[\cdot,\cdot]$ simply superposes the two sequences using the bijection β. The two functions are clearly bijections; moreover, given four sequences a,b,c,d, for even indexes we get

$$\langle[a,b],[c,d]\rangle_{2k} = [a,b]_k = \beta(a_k,b_k) = \beta(\langle a,c\rangle_{2k},\langle b,d\rangle_{2k}) = [\langle a,c\rangle,\langle b,d\rangle]_{2k},$$

and similarly for odd indexes, which proves that the two bijections are indeed companions.

In the rest of the section, we take an infinite set $\mathcal{D}$ and define a sufficient condition that a bijection $\mathcal{D}\times\mathcal{D}\to\mathcal{D}$ must satisfy in order for a companion bijection to exist. The result is not strictly needed for the development of a denotational semantics for the symmetric combinators (the examples given above suffice), so the uninterested reader may safely skip to Sect. 3.1.2.

In what follows, *tree* will always mean non-empty finite binary tree. Given two trees τ_1,τ_2, we denote $\tau_1\bullet\tau_2$ the tree having τ_1 and τ_2 as immediate subtrees, and $\tau_1\vee\tau_2$ their lub.

Obviously, $(\tau_1\bullet\tau_2)\vee(\tau_3\bullet\tau_4) = (\tau_1\vee\tau_3)\bullet(\tau_2\vee\tau_4)$. Notice that this equivalence is structurally identical to the equation defining companion bijections (Definition 3.1); unfortunately, $\bullet$ is not surjective, and $\vee$ is clearly not injective. As we will see, our solution will add some structure to binary trees to make these two functions "become" bijective.

In the following, $\mathcal{D}^2 = \mathcal{D}\times\mathcal{D}$.

Definition 3.2 (Fixpoints) *Let $\langle\cdot,\cdot\rangle$ be a function from $\mathcal{D}^2$ to $\mathcal{D}$. A* full fixpoint *of $\langle\cdot,\cdot\rangle$ is an element $p\in\mathcal{D}$ such that $\langle p,p\rangle = p$. A* fixpoint *of $\langle\cdot,\cdot\rangle$ is an element $h\in\mathcal{D}$ such that $\langle h,p\rangle = h$ for some full fixpoint p (therefore, full fixpoints are particular fixpoints).*

Definition 3.3 (Finitary bijection) *Let $\langle\cdot,\cdot\rangle:\mathcal{D}^2\to\mathcal{D}$ be a bijection, and let $\mathcal{F}$ be the set of its fixpoints. If $\mathcal{F}$ generates $\mathcal{D}$ through $\langle\cdot,\cdot\rangle$, then $\langle\cdot,\cdot\rangle$ is said to be* finitary.

If $\langle\cdot,\cdot\rangle$ is a finitary bijection, any element $a \in \mathcal{D}$ can be seen as a tree whose leaves are labelled by fixpoints of $\langle\cdot,\cdot\rangle$. More generally, given a tree τ, we say that a *labeling* for τ is a function from the leaves of τ to the fixpoints of $\langle\cdot,\cdot\rangle$. We will use the notation $\tau\{f\}$ to indicate that the tree τ is labelled by f. The element of $\mathcal{D}$ corresponding to $\tau\{f\}$ will be called its *value*, and denoted by $\ll \tau\{f\} \gg$.

If f_1, f_2 are two functions of disjoint domains X_1, X_2 to $\mathcal{D}$, we write $f_1 \bullet f_2$ for the function of domain $X_1 \cup X_2$ to $\mathcal{D}$ mapping x to $f_i(x)$ whenever $x \in X_i$. Using this notation, it is clear that if $\tau_1\{f_1\}$ and $\tau_2\{f_2\}$ are two labelled trees, then $\tau_1 \bullet \tau_2\{f_1 \bullet f_2\}$ is also a well defined labelled tree. Moreover, we have

Lemma 3.1 $\ll \tau_1 \bullet \tau_2\{f_1 \bullet f_2\} \gg = \langle \ll \tau_1\{f_1\} \gg, \ll \tau_2\{f_2\} \gg \rangle$.

Proof. Straightforward. □

Let f_1, f_2 be two functions from a set X to $\mathcal{D}$, and g a function from $\mathcal{D}^2$ to $\mathcal{D}$. Then, we write $g(\!|f_1, f_2|\!)$ for the function from X to $\mathcal{D}$ that maps $x \in X$ to $g(f_1(x), f_2(x))$.

Lemma 3.2 *Let $f_1, f_3 : X \to \mathcal{D}$ and $f_2, f_4 : Y \to \mathcal{D}$ be four functions such that $X \cap Y = \emptyset$, and $g : \mathcal{D}^2 \to \mathcal{D}$. Then, $g(\!|f_1 \bullet f_2, f_3 \bullet f_4|\!) = g(\!|f_1, f_3|\!) \bullet g(\!|f_2, f_4|\!)$.*

Proof. Straightforward. □

Labelled trees having the same value are naturally order by $\tau\{f\} \leq \tau'\{f'\}$ iff $\tau \leq \tau'$, so we can introduce the following definition:

Definition 3.4 (Canonical tree) *If $\langle\cdot,\cdot\rangle : \mathcal{D}^2 \to \mathcal{D}$ is a finitary bijection, the* canonical tree *of $a \in \mathcal{D}$ is the smallest element of the set*

$$\{\tau\{f\} \mid \ll \tau\{f\} \gg = a\} \ .$$

Proposition 3.3 (Extension) *Let $\tau\{f\}$ be the canonical tree of $a \in \mathcal{D}$, and τ' a tree such that $\tau < \tau'$. Then, there exists a unique labeling $\widehat{f}$ of τ', called the* extension *of f w.r.t. τ', such that $\ll \tau'\{\widehat{f}\} \gg = a$.*

Proof. $\widehat{f}$ clearly exists, it suffices to extend f remembering that a fixpoint h is such that $\langle h, p\rangle = h$. The uniqueness comes from the fact that $\langle\cdot,\cdot\rangle$ is a bijection. □

Lemma 3.4 *Let $\tau\{f_1 \bullet f_2\}$ be the canonical tree of $a \in \mathcal{D}$, and let $\tau < \tau'$. Then, the extension of $f_1 \bullet f_2$ with respect to τ' is such that*

$$\widehat{f_1 \bullet f_2} = \widehat{f_1} \bullet \widehat{f_2} \ .$$

Proof. We first observe that if the labeling is of the form $f_1 \bullet f_2$, then $\tau = \tau_1 \bullet \tau_2$ as well, and by definition we have $\ll \tau_i\{f_i\} \gg = a_i$, with $\langle a_1, a_2\rangle = a$. Now, if $\tau_1 \bullet \tau_2 < \tau'$, then it is necessarily $\tau' = \tau'_1 \bullet \tau'_2$, with $\tau_i \leq \tau'_i$. It is clear then that if $\widehat{f_i}$ are the extensions verifying $\ll \tau'_i\{\widehat{f_i}\} \gg = a_i$, then $\ll \tau'_1 \bullet \tau'_2\{\widehat{f_1} \bullet \widehat{f_2}\} \gg = a$, and we conclude by the uniqueness of the extension (Proposition 3.3). □

As remarked above, a finitary bijection $\langle\cdot,\cdot\rangle$ can be seen as a bijective version of $\bullet$ (the binary tree constructor). As a matter of fact, the only tree which cannot be written as $\tau_1 \bullet \tau_2$ is the tree made up of a single root/leaf x. Now,

if $h = \langle h, p\rangle$ (with p full fixpoint), we have that the canonical trees of h and p are resp. $x\{f\}$ and $x'\{f'\}$, where $f(x) = h$ and $f(x') = p$. We clearly have $\ll x \bullet x'\{f \bullet f'\} \gg = h$, so even single-node trees can be seen as composition of two subtrees. We now need a way to make the lub function bijective too; this will be possible when $\langle\cdot,\cdot\rangle$ is "rich" enough:

Definition 3.5 (Rich bijection) *A bijection between $\mathcal{D}^2$ and $\mathcal{D}$ is said to be* rich *iff it has an infinite number of fixpoints.*

In the sequel we shall prove that being finitary *and* rich is a sufficient condition for a bijection to admit a companion:

Theorem 3.5 *A finitary rich bijection admits a companion.*

Suppose the existence of a finitary rich bijection $\langle\cdot,\cdot\rangle : \mathcal{D}^2 \to \mathcal{D}$. We call $\mathcal{F}$ the set of its fixpoints. Since $\langle\cdot,\cdot\rangle$ is rich, there exists a bijection $\beta : \mathcal{F}^2 \to \mathcal{F}$. Now let $a, b \in \mathcal{D}$, let $\tau_a\{f_a\}$ and $\tau_b\{f_b\}$ be their respective canonical trees, and let $\widehat{f_a}$ and $\widehat{f_b}$ be the extensions of the two labellings w.r.t. $\tau_a \vee \tau_b$. We define

$$[a, b] = \ll \tau_a \vee \tau_b\{\beta(\!|\widehat{f_a}, \widehat{f_b}|\!)\} \gg$$

$[\cdot,\cdot]$ is clearly a function from $\mathcal{D}^2$ to $\mathcal{D}$. We claim that $[\cdot,\cdot]$ is a bijection, indeed a companion for $\langle\cdot,\cdot\rangle$.

Let us first prove that $\langle\cdot,\cdot\rangle$ and $[\cdot,\cdot]$ satisfy the equation of Definition 3.1. Let $a_1, a_2, a_3, a_4 \in \mathcal{D}$, and let $\tau_i\{f_i\}$ be the canonical trees of a_i, for $1 \le i \le 4$. Then we have, thanks to Lemmas 3.1, 3.2, and 3.4,

$$\begin{aligned}
\langle [a_1, a_2], [a_3, a_4]\rangle &= \langle \ll \tau_1 \vee \tau_2\{\beta(\!|\widehat{f_1}, \widehat{f_2}|\!)\} \gg, \ll \tau_3 \vee \tau_4\{\beta(\!|\widehat{f_3}, \widehat{f_4}|\!)\} \gg\rangle = \\
&= \ll (\tau_1 \vee \tau_2) \bullet (\tau_3 \vee \tau_4)\{\beta(\!|\widehat{f_1}, \widehat{f_2}|\!) \bullet \beta(\!|\widehat{f_3}, \widehat{f_4}|\!)\} \gg \; = \\
&= \ll (\tau_1 \bullet \tau_3) \vee (\tau_2 \bullet \tau_4)\{\beta(\!|\widehat{f_1} \bullet \widehat{f_3}, \widehat{f_2} \bullet \widehat{f_4}|\!)\} \gg \; = \\
&= \ll (\tau_1 \bullet \tau_3) \vee (\tau_2 \bullet \tau_4)\{\beta(\!|\widehat{f_1 \bullet f_3}, \widehat{f_2 \bullet f_4}|\!)\} \gg \; = \\
&= [\ll \tau_1 \bullet \tau_3\{f_1 \bullet f_3\} \gg, \ll \tau_2 \bullet \tau_4\{f_2 \bullet f_4\} \gg] = \\
&= [\langle \tau_1\{f_1\}, \tau_3\{f_3\}\rangle, \langle \tau_2\{f_2\}, \tau_4\{f_4\}\rangle] = \\
&= [\langle a_1, a_3\rangle, \langle a_2, a_4\rangle]
\end{aligned}$$

All that remains to do is prove that $[\cdot,\cdot]$ is a bijection. First of all, let $\alpha_1, \alpha_2 : \mathcal{F} \to \mathcal{F}$ be the projections associated to β. We extend these projections to two functions $\rho_1, \rho_2 : \mathcal{D} \to \mathcal{D}$ by posing

$$\rho_i(a) = \begin{cases} \alpha_i(a) & \text{if } a \in \mathcal{F} \\ \langle \rho_i(a_1), \rho_i(a_2)\rangle & \text{if } a \notin \mathcal{F} \text{ and } a = \langle a_1, a_2\rangle \end{cases}$$

We will see that actually ρ_1 and ρ_2 are the projections associated to $[\cdot,\cdot]$.

Define the function $P : \mathcal{D} \to \mathcal{D}^2$ as $P(a) = (\rho_1(a), \rho_2(a))$. We will prove that P is the inverse of $[\cdot,\cdot]$, i.e., that

$$\begin{aligned}
\forall a \in \mathcal{D},\ [\rho_1(a), \rho_2(a)] &= a \\
\forall (a_1, a_2) \in \mathcal{D}^2,\ (\rho_1([a_1, a_2]), \rho_2([a_1, a_2])) &= (a_1, a_2)
\end{aligned}$$

which is enough to prove that $[\cdot,\cdot]$ is bijective.

Both equalities will be established by an induction on (a pair of) canonical trees. For the first one, start assuming that $a \in \mathcal{F}$, so that the canonical tree of a is just a leaf labelled by a itself, a situation that we denote by $\epsilon\{a\}$. Clearly, $\ll \epsilon\{a\} \gg= a$. So we have

$$[\rho_1(a), \rho_2(a)] = [\alpha_1(a), \alpha_2(a)] = \ll \epsilon \vee \epsilon\{\beta(\alpha_1(a), \alpha_2(a))\} \gg \; = \ll \epsilon\{a\} \gg$$

and the base case is verified. For the induction step, assume $a = \langle a_1, a_2 \rangle$. Then, we have

$$\begin{aligned}[\rho_1(\langle a_1, a_2\rangle), \rho_2(\langle a_1, a_2\rangle)] &= [\langle \rho_1(a_1), \rho_1(a_2)\rangle, \langle \rho_2(a_1), \rho_2(a_2)\rangle] = \\ &= \langle [\rho_1(a_1), \rho_2(a_1)], [\rho_1(a_2), \rho_2(a_2)]\rangle\end{aligned}$$

and we conclude using the induction hypothesis.

For the second equality, start with both $a_1, a_2 \in \mathcal{F}$. Then we have

$$\rho_i([a_1, a_2]) = \rho_i(\ll \epsilon\{\beta(a_1, a_2)\} \gg) = \alpha_i(\beta(a_1, a_2)) = a_i \ .$$

Then we proceed supposing that $a_1 \in \mathcal{F}$, but $a_2 = \langle a_1', a_2' \rangle$. Remembering that, being a_1 a fixpoint of $\langle \cdot, \cdot \rangle$, we have $\langle a_1, p \rangle = a_1$, we can write

$$\begin{aligned}\rho_i([a_1, \langle a_1', a_2'\rangle]) &= \rho_i([\langle a_1, p\rangle, \langle a_1', a_2'\rangle]) = \\ &= \rho_i(\langle [a_1, a_1'], [p, a_2']\rangle) = \\ &= \langle \rho_i([a_1, a_1']), \rho_i([p, a_2'])\rangle\end{aligned}$$

a_1 and p are both fixpoints, so we can apply the induction hypothesis to obtain

$$\begin{aligned}\rho_1([a_1, \langle a_1', a_2'\rangle]) &= \langle a_1, p\rangle = a_1 \\ \rho_2([a_1, \langle a_1', a_2'\rangle]) &= \langle a_1', a_2'\rangle\end{aligned}$$

It remains the case $a_1 = \langle a_1^1, a_2^1 \rangle$ and $a_2 = \langle a_1^2, a_2^2 \rangle$, for which we have

$$\rho_i([\langle a_1^1, a_2^1\rangle, \langle a_1^2, a_2^2\rangle]) = \rho_i(\langle [a_1^1, a_1^2], [a_2^1, a_2^2]\rangle) = \langle \rho_i([a_1^1, a_1^2]), \rho_i([a_2^1, a_2^2])\rangle$$

which by induction hypothesis yields $\langle a_1^i, a_2^i \rangle$.

We remark here that being finitary and rich is not a necessary condition for a bijection to admit a companion: as a matter of fact, the set $\Phi_2(\mathcal{A})$ defined at p. 107 admits two finitary companion bijections which are not rich as soon as the base set $\mathcal{A}$ is finite.

3.1.2 Experiments and interpretation

The previous section suggests the following definition:

Definition 3.6 (Interaction set) *An* interaction set *is an infinite pointed set $\mathcal{D}$ (the distinguished element being denoted by $\mathbf{0}$), admitting two companion bijections*

$$\langle \cdot, \cdot \rangle, [\cdot, \cdot] : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$$

such that $\langle \mathbf{0}, \mathbf{0} \rangle = [\mathbf{0}, \mathbf{0}] = \mathbf{0}$.

The distinguished element will be used to interpret the ε combinator. Before going on, let us prove with a few examples that interaction sets exist. Let $\Phi(\mathbb{N})$ be the set of almost everywhere-zero sequences of natural numbers, and let β be Cantor's bijection between $\mathbb{N} \times \mathbb{N}$ and $\mathbb{N}$, i.e., if $m, n \in \mathbb{N}$,

$$\beta(m,n) = \frac{(m+n+1)(m+n)}{2} + m.$$

The distinguished element $\mathbf{0}$ will be the everywhere-zero sequence. Since $\Phi(\mathbb{N})$ is a subset of $\mathcal{S}(\mathbb{N})$, we can define the two companion bijections $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$ of p. 103. It is obvious that $\langle \mathbf{0}, \mathbf{0} \rangle = \mathbf{0}$; on the other hand, the equality $[\mathbf{0}, \mathbf{0}] = \mathbf{0}$ is assured by the fact that $\beta(0,0) = 0$. The restriction to almost everywhere-zero sequences is not fundamental; we imposed it just to show that denumerable interaction sets exist.

The following is a more symmetric example, which does not rely on a specific bijection β; indeed, it does not even require such a bijection to exist, as the base set $\mathcal{A}$ can be finite. Take any set $\mathcal{A}$ with at least two elements, one of them denoted by 0, and define $\Phi_2(\mathcal{A})$ to be the set of almost everywhere-zero sequences of elements of $\mathcal{A}$ indexed by pairs of natural numbers. For any $x, y \in \Phi_2(\mathcal{A})$, we can define two bijections between $\Phi_2(\mathcal{A}) \times \Phi_2(\mathcal{A})$ and $\Phi_2(\mathcal{A})$ as follows:

$$\langle x, y \rangle_{i,j} = \begin{cases} x_{k,j} & \text{if } i = 2k \\ y_{k,j} & \text{if } i = 2k+1 \end{cases}$$

$$[x, y]_{i,j} = \begin{cases} x_{i,k} & \text{if } i = 2k \\ y_{i,k} & \text{if } i = 2k+1 \end{cases}$$

The fact that the functions just defined are bijective is obvious: they just build a new sequence by interleaving two sequences; one of them interleaves them "horizontally", the other "vertically". In other words, our sequences being bidimensional, i.e., "sequences of sequences", in the first case we consider them as "sequences of columns", and interleave them horizontally; in the second case, we consider them as "sequences of rows", and interleave them vertically. We leave it to the reader to check that $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$ are companions; the fact that $\langle \mathbf{0}, \mathbf{0} \rangle = [\mathbf{0}, \mathbf{0}] = \mathbf{0}$ (where $\mathbf{0}$ is the everywhere-zero sequence) is obvious.

In Sect. 3.3 we shall indeed show that, when $\mathcal{A}$ is a monoid, $\Phi_2(\mathcal{A})$ with the above bijections is an *interaction monoid*, so in particular an interaction set. Interaction sets of the form $\Phi_2(\mathcal{A})$ will be considered in Sect. 3.2 to show the existence of models which are fully abstract with respect to the observational equivalence defined in Sect. 2.1. Even though $\mathcal{S}_2(\mathcal{A})$ (the set of *all* bidimensional sequences on $\mathcal{A}$) is also an interaction set, such a result will crucially depend on the fact that the sequences considered are almost everywhere-zero.

Let us now see how one can interpret nets of symmetric combinators into a generic interaction set $\mathcal{D}$.

Definition 3.7 (Experiment) *Let μ be a net. An* experiment *on μ is a function $e : \mathsf{Ports}(\mu) \to \mathcal{D}$ such that:*

(a) if $i, j \in \mathsf{Ports}(\mu)$ are connected by a wire, then $e(i) = e(j)$;

(b) if $i, j, k \in \mathsf{Ports}(\mu)$ are resp. the left auxiliary, right auxiliary, and principal port of a δ cell of μ, then $e(k) = \langle e(i), e(j) \rangle$;

(c) *if* $i, j, k \in \mathsf{Ports}(\mu)$ *are resp. the left auxiliary, right auxiliary, and principal port of a* ζ *cell of* μ*, then* $e(k) = [e(i), e(j)]$*;*

(d) *if* $i \in \mathsf{Ports}(\mu)$ *is the principal port of an* ε *cell, then* $e(i) = \mathbf{0}$*.*

If $k_1, \ldots, k_n$ *are the free ports of* μ*, with* $n \geq 1$*, the tuple* $(e(k_1), \ldots, e(k_n))$ *is called the* result *of the experiment and is denoted by* $|e|$*.*

In the following, we write $\mathcal{D}^n$ for $\mathcal{D} \times \cdots \times \mathcal{D}$ n times.

Definition 3.8 (Interpretation) *Let* μ *be a net with* $n \geq 1$ *free ports. The* interpretation *of* μ *in* $\mathcal{D}$*, written* $[\![\mu]\!]$*, is defined to be the subset of* $\mathcal{D}^n$ *containing the results of all possible experiments on* μ*:*

$$[\![\mu]\!] = \{|e| \ ; \ e \text{ experiment on } \mu\}.$$

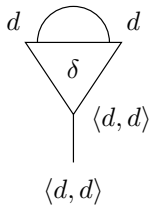
The interpretation of a net is always a pointed set, i.e., it is never empty. This is an immediate consequence of the definition of experiment and of the fact that $\mathbf{0}$ is a fixpoint of both bijections:

Proposition 3.6 *For all* μ *with at least one free port,* $\mathbf{0} \in [\![\mu]\!]$*.*

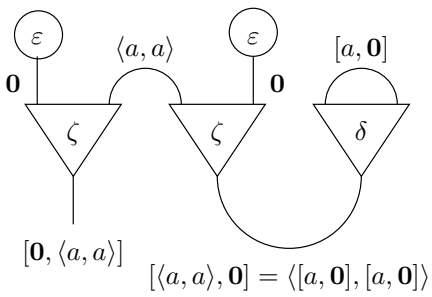
Proof. The function assigning $\mathbf{0}$ to all ports of μ is an experiment. □

We can give a few examples to see some concrete applications of the above definition. Consider the package ε consisting of a single ε cell. There is only one possible experiment on it, which assigns $\mathbf{0}$ to the principal port of the ε cell and to the free port of the package, so $[\![\varepsilon]\!] = \{\mathbf{0}\}$. By Proposition 3.6, this is the net with the smallest interpretation.

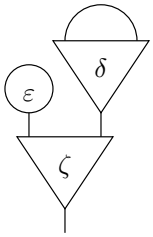
If we take the package δ consisting of a single δ cell whose auxiliary ports are connected by a wire, we clearly have that all possible experiments are of the following form:



so $[\![\delta]\!] = \{\langle d, d\rangle \ ; \ d \in \mathcal{D}\}$. Just as an axiom in linear logic, the net W with two free ports consisting of a single wire is interpreted by the diagonal relation in $\mathcal{D} \times \mathcal{D}$: $[\![W]\!] = \{(d, d) \ ; \ d \in \mathcal{D}\}$. The following is a more involved example:



In the above picture, a label d on a wire means that the two ports connected by the wire have both been assigned the element d by the experiment; a is a generic element of $\mathcal{D}$. We therefore see that, if we call μ the above net, we have $[\![\mu]\!] = \{[\mathbf{0}, \langle a, a\rangle] \ ; \ a \in \mathcal{D}\}$. The reader can check that this is also the interpretation of the following net



which is the cut-free of form of μ.

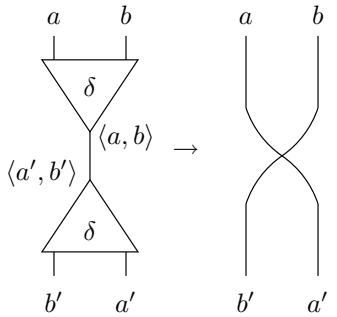
As a matter of fact, the interpretation is a denotational semantics for the symmetric combinators, i.e., it is preserved under reduction. Additionally, it also models $\simeq_\eta$.

Lemma 3.7 (Stability under reduction) *Let μ, μ' be two nets with at least one free port. Then, $\mu \to \mu'$ implies $[\![\mu]\!] = [\![\mu']\!]$.*

Proof. We need to show that for any experiment e on μ, there exists an experiment e' on μ' yielding the same result, and vice-versa. Since the rewriting is local, it actually suffices to show that, for all reduction rules, the assignment given by the experiment e on the interface of the left member of the rule can be reproduced by e' on the interface of the right member, and vice-versa; at this point e and e' can be assumed to be equal everywhere else, which guarantees that the results are the same.

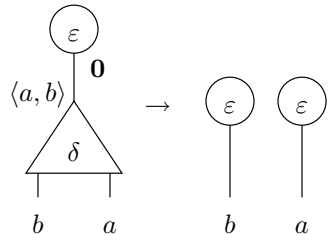
The case of the $\varepsilon\varepsilon$ annihilation is trivial: e' is just e restricted to the ports which do not disappear after the application of the rule.

The $\delta\delta$ and $\zeta\zeta$ annihilations are structurally identical, so we shall only consider the first one:



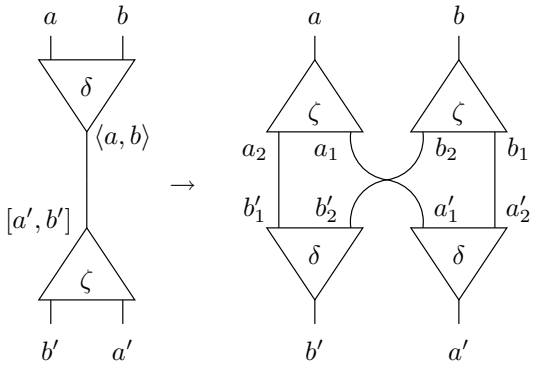
Here, a, b, a', b' are generic elements of $\mathcal{D}$. The assignment on the left hand side must satisfy $\langle a, b\rangle = \langle a', b'\rangle$, which by the injectivity of $\langle \cdot, \cdot \rangle$ implies $a = a'$ and $b = b'$, therefore the assignment on the right hand side is correct. The converse is trivial.

For what concerns the commutations, the $\delta\varepsilon$ and $\zeta\varepsilon$ commutations are again structurally identical, so we only need to consider the first one:



Again, by injectivity of $\langle\cdot,\cdot\rangle$, the requirement on the left hand side that $\langle a, b\rangle = \mathbf{0}$ implies $a = \mathbf{0}$ and $b = \mathbf{0}$, so the assignment on the right hand side is correct. The converse holds because of the hypothesis that $\langle \mathbf{0}, \mathbf{0}\rangle = \mathbf{0}$.

On the other hand, for the $\delta\zeta$ commutation, we get



In the left hand side we must have $\langle a, b\rangle = [a', b']$. By surjectivity of $\langle\cdot,\cdot\rangle$ and $[\cdot,\cdot]$, there exist $a_1, a_2, a'_1, a'_2, b_1, b_2, b'_1, b'_2 \in \mathcal{D}$ such that $a = [a_1, a_2]$, $b = [b_1, b_2]$, $a' = \langle a'_1, a'_2\rangle$, and $b' = \langle b'_1, b'_2\rangle$. The above equality and the fact that $\langle\cdot,\cdot\rangle$ and $[\cdot,\cdot]$ are companions imply $[\langle a'_1, a'_2\rangle, \langle b'_1, b'_2\rangle] = [\langle a_1, b_1\rangle, \langle a_2, b_2\rangle]$, which by injectivity of $\langle\cdot,\cdot\rangle$ and $[\cdot,\cdot]$ in turn implies $a'_1 = a_1$, $a'_2 = b_1$, $b'_1 = a_2$, and $b'_2 = b_2$. Therefore, the assignment defined above for the right hand side of the rule is correct. Conversely, if we know from the right hand side that $a'_1 = a_1 = c_1$, $a'_2 = b_1 = c_2$, $b'_1 = a_2 = c_3$, and $b'_2 = b_2 = c_4$, we have $a = [c_1, c_3]$, $b = [c_2, c_4]$, $a' = \langle c_1, c_2\rangle$, and $b' = [c_3, c_4]$, which means that $\langle a, b\rangle = \langle [c_1, c_3], [c_2, c_4]\rangle$ and $[a', b'] = [\langle c_1, c_2\rangle, \langle c_3, c_4\rangle]$. But since $\langle\cdot,\cdot\rangle$ and $[\cdot,\cdot]$ are companions, this implies that $\langle a, b\rangle = [a', b']$, so the assignment on the left hand side is correct. □

Lemma 3.8 (Congruence) *Let μ, μ' be two nets with n free ports, and let C be a context for μ, μ'. Then, $[\![\mu]\!] = [\![\mu']\!]$ implies $[\![C[\mu]]\!] = [\![C[\mu']]\!]$.*

Proof. An experiment on $C[\mu]$ must be the union of an experiment e on μ and an experiment f on C, such that, if i and j are two free ports of resp. μ and C which are connected in $C[\mu]$, $e(i) = f(j)$. The same holds for $C[\mu']$, so, if m is the number of free ports of $C[\mu]$ and $C[\mu']$, we have

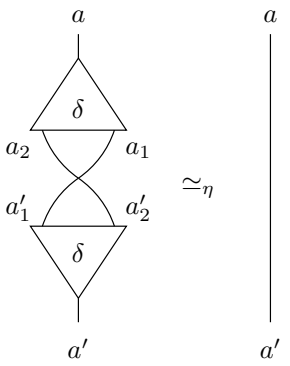
$$[\![C[\mu]]\!] = \{d \in \mathcal{D}^m \ ; \ (d, c) \in [\![C]\!] \text{ and } c \in [\![\mu]\!]\}$$

$$[\![C[\mu']]\!] = \{d' \in \mathcal{D}^m \ ; \ (d', c) \in [\![C]\!] \text{ and } c \in [\![\mu']\!]\},$$

from which we clearly see that if $[\![\mu]\!] = [\![\mu']\!]$, then $[\![C[\mu]]\!] = [\![C[\mu']]\!]$. □

Lemma 3.9 (Extensionality) *Let μ, μ' be two nets with at least one free port. Then, $\mu \simeq_\eta \mu'$ implies $[\![\mu]\!] = [\![\mu']\!]$.*

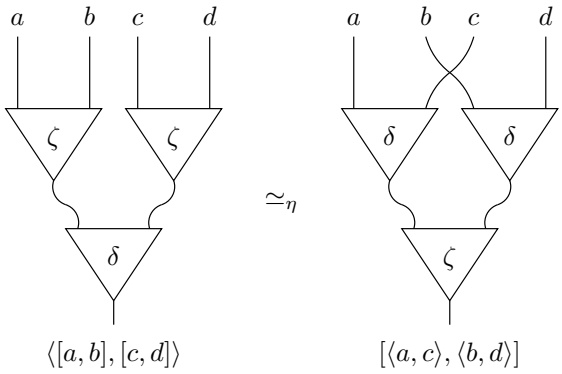
Proof. The proof follows exactly the same argument used for Lemma 3.7. We start by considering the η-expansion for δ, the corresponding rule involving ζ being structurally identical:



The left hand side imposes $a'_1 = a_1$ and $a'_2 = a_2$, which implies $a = a'$. Conversely, the right hand side imposes $a'_1 = a_1$; by surjectivity of $\langle \cdot, \cdot \rangle$, a_1 and a_2 such that $\langle a_1, a_2 \rangle = a$ exist, so the assignment on the left hand side is correct.

The cases of the $\delta\varepsilon$ and $\zeta\varepsilon$ commutations are trivial, and rest upon the fact that $\langle \mathbf{0}, \mathbf{0} \rangle = [\mathbf{0}, \mathbf{0}] = \mathbf{0}$.

The case of the $\delta\zeta$ commutation is also trivial:



The two experiments are the same thanks to the fact that $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$ are companions. □

Lemmas 3.7 and 3.8 together prove that $[\![\cdot]\!]$ is a denotational semantics of $\simeq_\beta$ for the symmetric combinators: it is preserved under reduction, and it is a congruence. Lemma 3.9 proves that the semantics actually models $\simeq_{\beta\eta}$. Moreover, the examples given at p. 108 show that there exist two nets μ, μ' such that $[\![\mu]\!] \neq [\![\mu']\!]$, so the semantics is non-trivial.

3.1.3 Injectivity

So far, we know that for any two nets with at least one free port, $\mu \simeq_{\beta\eta} \mu'$ implies $[\![\mu]\!] = [\![\mu']\!]$. If we restrict to *total* nets, the internal separation result

stated in Sect. 2.1 (Theorem 2.19) ensures also the *injectivity* of the semantics with respect to $\simeq_{\beta\eta}$, i.e., if two total nets have the same interpretation, then they are $\beta\eta$-equivalent.

Theorem 3.10 (Injectivity) *Let μ, μ' be two total nets with at least one free port. Then, $\mu \simeq_{\beta\eta} \mu'$ iff $[\![\mu]\!] = [\![\mu']\!]$.*

Proof. As already said, the forward implication is a consequence of Lemmas 3.7 and 3.9. For the converse, which is the actual injectivity property, we can restrict to packages, since:

- total nets containing active pairs can be reduced and their cut-free form considered;
- cut-free nets with more than one free port can be "closed" by means of any fixed tree, and the argument below can thus be easily generalized.

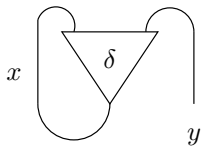
So take two packages π, π' such that $\pi \not\simeq_{\beta\eta} \pi'$. By Theorem 2.19, there exists a 2-test θ such that $\theta[\pi] \to^* E$ and $\theta[\pi'] \to^* W$ (or viceversa, but we do not lose generality in assuming this situation), where we rimind that E is the net with two free ports consisting consisting of two ε cells and W is a single wire. We therefore have

$$[\![\theta[\pi]]\!] = [\![E]\!] = \{(\mathbf{0}, \mathbf{0})\}$$

$$[\![\theta[\mu']]\!] = [\![W]\!] = \{(d, d) \ ; \ d \in \mathcal{D}\}.$$

Since $\mathcal{D}$ is infinite, $[\![\theta[\pi]]\!] \neq [\![\theta[\pi']]\!]$, which by Lemma 3.8 implies $[\![\pi]\!] \neq [\![\pi']\!]$. □

Injectivity with respect to $\simeq_{\beta\eta}$ fails in general for non-total nets, as the following example shows. Let $\Phi_2(\mathcal{A})$ be an interaction set as introduced at p. 107. Now consider the following non-total net μ:



The labels indicate that the generic experiment on μ in $\Phi_2(\mathcal{A})$ assigns the sequence x on the left auxiliary and principal ports of the δ cell, and the sequence y to the right auxiliary port of the δ cell and to the free port of μ. Therefore, we have

$$[\![\mu]\!] = \{y \in \Phi_2(\mathcal{A}) \ ; \ \exists x \in \Phi_2(\mathcal{A}).\langle x, y\rangle = x\}.$$

But Lemma 3.12 (see Sect. 3.2 below) proves that the only possible such y is $\mathbf{0}$, so $[\![\mu]\!] = \{\mathbf{0}\} = [\![\varepsilon]\!]$, where ε is the package consisting of the sole ε combinator. Now, both μ and ε do not contain active pairs, hence the only hope of rewriting one into the other is through η-equivalence. But a simple inspection of the η-rules of Definition 2.19 reveals that the presence of ε cells is preserved by $\simeq_\eta$: no rule can produce a net containing no ε cell from a net containing one, and no rule can add ε cells if there are none. Therefore, $\mu \not\simeq_{\beta\eta} \varepsilon$, and yet $[\![\mu]\!] = [\![\varepsilon]\!]$.

This is actually not so surprising; when non-total nets are considered, the situation is in some sense similar to considering non-normalizable terms in the λ-calculus. For example, Ω and $\lambda x.\Omega$ are not $\beta\eta$-equivalent, and yet they have

the same interpretation in any sensible model, since they are both unsolvable. Indeed, we shall see that the interaction sets of the form $\Phi_2(\mathcal{A})$ are fully abstract models of the observational equivalence defined in Sect. 2.1; in this sense, identifying the net μ above with the ε combinator is observationally sound, i.e., $\mu \simeq \varepsilon$.

3.1.4 Full completeness

If $\mathcal{D}$ is an interaction set, even denumerable, for obvious resons of cardinality not every subset of $\mathcal{D}^n$ is the interpretation of some net. In this section we characterize those that are interpretations of *total* nets.

In the following, $\mathcal{D}$ is a generic interaction set.

Definition 3.9 (Bracket expression) *Let x range over a denumerable set of variables. A* simple bracket expression b *is a syntactical expression belonging to the following grammar:*

$$\mathsf{b} ::= x \mid 0 \mid \langle \mathsf{b}, \mathsf{b} \rangle \mid [\mathsf{b}, \mathsf{b}]$$

A bracket expression *is a tuple of simple bracket expressions.*

We denote by $\mathsf{var}(\mathsf{b})$ the set of variables occurring in the simple bracket expression b. We define as usual the substitution of a variable y in place of x in b, denoted by $\mathsf{b}[y/x]$. If $x \in \mathsf{var}(\mathsf{b})$ and $z \notin \mathsf{var}(\mathsf{b})$, then we say that b and $\mathsf{b}[x/z]$ are α-equivalent.

Similarly, if $\mathsf{B} = (\mathsf{b}_1, \ldots, \mathsf{b}_n)$ is a bracket expression, we define $\mathsf{var}(\mathsf{B}) = \mathsf{var}(\mathsf{b}_1) \cup \cdots \cup \mathsf{var}(\mathsf{b}_n)$, i.e., variables are shared by the simple expressions in the tuple, and substitution is performed on the whole expression; α-equivalence is trivially extended, and bracket expressions are always considered modulo α-equivalence.

If B is a bracket expression containing n simple expressions such that $\mathsf{var}(\mathsf{B}) \subseteq \{x_1, \ldots, x_m\}$, and if $d_1, \ldots d_m \in \mathcal{D}$, we can define an element $\mathsf{B}\{x_1 := d_1, \ldots, x_m := d_m\}$ of $\mathcal{D}^n$ in the obvious way: just assign each d_i to x_i, and compute the expression considering the symbols 0, $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$ as resp. the distinguished element and the two bijections of $\mathcal{D}$. For example, suppose that d, e, f are three elements of $\mathcal{D}$ such that $f = [\langle d, e \rangle, d]$; then, if $\mathsf{B} = [\langle x, y \rangle, x]$, we have $\mathsf{B}\{x := d, y := e\} = f$. In this way, each bracket expression B containing n simple bracket expressions and a total of m variables defines a function from $\mathcal{D}^m$ to $\mathcal{D}^n$. Because of the obvious shortage of bracket expressions, the assignment cannot be surjective; it is not injective either, as the expressions 0 and $\langle 0, 0 \rangle$ show (they both represent the constant function $\mathbf{0}$).

In the following, $\mathsf{occ}_x(\mathsf{B})$ denotes the number of occurrences of the variable x in the bracket expression B.

Definition 3.10 (Balanced bracket expression) *A bracket expression B is* balanced *iff, for any variable x, either $\mathsf{occ}_x(\mathsf{B}) = 0$ or $\mathsf{occ}_x(\mathsf{B}) = 2$. A function from $\mathcal{D}^m$ to $\mathcal{D}^n$ is said to be* balanced *if it can be defined through a balanced bracket expression containing n expressions using m variables. A set $\mathcal{B} \subseteq \mathcal{D}^n$ is called* balanced *if it is the codomain of a balanced function.*

Theorem 3.11 (Full completeness) *If μ is a total net with $n \geq 1$ free ports, then $[\![\mu]\!]$ is balanced. Conversely, if $n \geq 1$, given a balanced set $\mathcal{B} \subseteq \mathcal{D}^n$, there exists a cut-free net μ with n free ports such that $[\![\mu]\!] = \mathcal{B}$.*

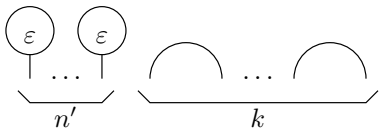
Proof. Since μ is total, we can consider its cut-free form ν. By Lemma 3.7, $[\![\mu]\!] = [\![\nu]\!]$; now, if we remember that cut-free nets are trees of δ and ζ cells with wires and ε cells "on top", it is clear that the first statement is a straight-forward consequence of the definition of experiments.

For what concerns the converse, let $\mathcal{B} \subseteq \mathcal{D}^n$ be balanced, and let B be the bracket expression such that $\mathcal{B}$ is the codomain of B. Simple bracket expressions can obviously be provided with a complexity measure $\sharp(\cdot)$, which is the total number of binary syntactical constructs used:

- $\sharp(x) = \sharp(\mathsf{0}) = 0$;
- $\sharp(\langle \mathsf{b}_1, \mathsf{b}_2 \rangle) = \sharp([\mathsf{b}_1, \mathsf{b}_2]) = \sharp(\mathsf{b}_1) + \sharp(\mathsf{b}_2) + 1$.

For an expression $\mathsf{B} = (\mathsf{b}_1, \ldots, \mathsf{b}_n)$, we pose $\sharp(\mathsf{B}) = \sharp(\mathsf{b}_1) + \cdots + \sharp(\mathsf{b}_n)$.

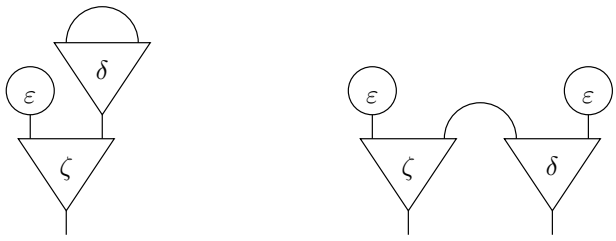
We can then reason by induction on $\sharp(\mathsf{B})$. If $\sharp(\mathsf{B}) = 0$, knowing that B is balanced, we can assume without loss of generality that $\mathsf{B} = (\mathsf{0}, \ldots, \mathsf{0}, x_1, x_1, \ldots, x_k, x_k)$. It is easy to see that if we interpret the net



(where $n' + k = n$), we obtain exactly $\mathcal{B}$.

If $\sharp(\mathsf{B}) > 0$, then we can assume without loss of generality that $\mathsf{B} = (\langle \mathsf{b}'_1, \mathsf{b}''_1 \rangle, \mathsf{b}_2, \ldots, \mathsf{b}_n)$ or $\mathsf{B} = ([\mathsf{b}'_1, \mathsf{b}''_1], \mathsf{b}_2, \ldots, \mathsf{b}_n)$. In both cases, the bracket expression $\mathsf{B}' = (\mathsf{b}'_1, \mathsf{b}''_1, \mathsf{b}_2, \ldots, \mathsf{b}_n)$ has measure strictly smaller than B, so the induction hypothesis applies, giving us a net μ' with $n+1$ free ports such that $[\![\mu']\!]$ is the codomain of B'. Clearly, adding a δ or a ζ cell (according to the shape of B) to μ' yields a net with n free ports μ such that $[\![\mu]\!] = \mathcal{B}$. □

The above proof actually tells us that, for $n \geq 1$, the balanced subsets of $\mathcal{D}^n$ are in bijection with the cut-free nets with n free ports. As an example, take the balanced sets induced by the expressions $[\mathsf{0}, \langle x, x \rangle]$ and $([\mathsf{0}, x], \langle x, \mathsf{0} \rangle)$; they correspond to the following two nets:



This is not surprising at all: a balanced expression is just a list of trees, the connections between leaves being expressed as pairs of occurrences of the same variable. Therefore, balanced expressions are nothing but an alternative, linear syntax for cut-free nets. It must also be mentioned that this full completeness result is very similar to that proved by Michele Pagani for multiplicative linear logic proof-nets [Pag06].

3.1.5 Semantical characterization of observability

We now give a compact semantical characterization of observability in the model given by the interaction set $\Phi_2(\mathcal{A})$, introduced at p. 107 (where $\mathcal{A}$ is any set with at least two elements, one of them called 0). Namely, we shall prove that, in this model, a net is blind iff its interpretation is the smallest possible one, i.e., the set containing only the tuple $(\mathbf{0}, \ldots, \mathbf{0})$, which by Proposition 3.6 must always be part of the interpretation of a net. This is similar to what happens in sensible order-theoretical models of the λ-calculus, in which unsolvable terms are all assigned the least element of the domain.

In the following, we call *scalars* the elements of $\mathcal{A}$, and we identify the scalar a and the element $x \in \Phi_2(\mathcal{A})$ such that $x_{0,0} = a$ and $x_{i,j} = 0$ everywhere else.

Lemma 3.12 *Both $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$ are finitary (see Definition 3.3).*

Proof. First of all, we remark that, for any scalar a (considered as an element of $\Phi_2(\mathcal{A})$),

$$\langle a, \mathbf{0} \rangle = [a, \mathbf{0}] = a,$$

and that these are the only fixpoints of $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$. Now let $\| \cdot \| : \mathcal{A} \to \{0, 1\}$ be the discrete norm on $\mathcal{A}$, i.e., given a scalar a, $\|a\| = 0$ if a is zero, $\|a\| = 1$ otherwise. Then, for all $x \in \Phi_2(\mathcal{A})$ we define

$$\sharp x = \sum_{i,j} (i + j) \|x_{i,j}\|.$$

In other words, $\sharp x$ is the sum of the number of non-zero elements of x, weighed with respect to their distance from $(0, 0)$: the farther a non-zero element is found in the sequence, the higher its weight will be. In particular, if x is a scalar, $\sharp x = 0$.

Now let π_1, π_2 and ρ_1, ρ_2 be the left and right projections associated resp. to $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$. It is not hard to check that, for all $x \in \Phi_2(\mathcal{A})$ and for all $i \in \{1, 2\}$, $\sharp \pi_i(x) \leq \sharp x$, and $\sharp \rho_i(x) \leq \sharp x$, and that equality holds iff x is a scalar. Therefore, both $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$ can generate any element of $\Phi_2(\mathcal{A})$ starting from scalars, i.e., their fixpoints. □

The above result means that we can see the elements of $\Phi_2(\mathcal{A})$ as finite binary trees whose leaves are labelled by scalars (cf. Definition 3.4). In particular, if we pose

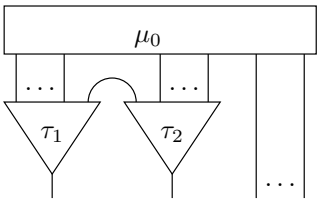
$$\mathcal{C} = \bigcup_{1 \leq k < \omega} \Phi_2(\mathcal{A})^k,$$

we can extend the measure $\sharp$ defined in the above proof to the elements of $\mathcal{C}$ as follows,

$$\text{for all } (x_1, \ldots, x_n) \in \mathcal{C},\ \sharp(x_1, \ldots, x_n) = \sum_{k=1}^{n} \sharp x_k.$$

This will allow us to reason by induction when considering tuples of elements of $\Phi_2(\mathcal{A})$.

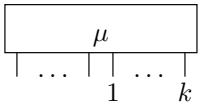
We shall now prove by a realizability argument (cf. Jean-Louis Krivine's book [Kri90]) that, given a net μ with $n \geq 1$ free ports, if $[\![\mu]\!]$ contains a non-zero tuple of $\Phi_2(\mathcal{A})^n$, then $\mu\Downarrow$. The converse is trivial; by definition, $\mu\Downarrow$ means that μ reduces to a net which without loss of generality can be assumed to be of the form



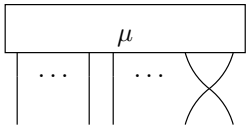
We can then consider the experiment assigning $\mathbf{0}$ to all ports of μ_0, and an element $x \neq \mathbf{0}$ to the two ports connected by the wire shown in the above picture. By injectivity of $\langle \cdot, \cdot \rangle$ and $[\cdot, \cdot]$, the two "leftmost" free ports (or just the "leftmost" free port if $\tau_1 = \tau_2$ and the wire connects two leaves of the same tree) are labelled by non-zero elements of $\Phi_2(\mathcal{A})$. Then, by invariance under reduction, $(x_1, x_2, \ldots, x_n) \in [\![\mu]\!]$, where at least $x_1 \neq \mathbf{0}$.

We shall now define what it means for a net to *realize* an element of $\mathcal{C}$. The difference with realizability in the λ-calculus is that we shall need to specify *on which ports* a net realizes a tuple, i.e., we shall introduce realization with respect to a certain subset of the interface of a net.

To make notations lighter, we shall use the following conventions. First of all, we assume that the free ports of nets are numbered by positive intergers. Then, whenever a net μ with $n \geq 1$ free ports realizes $(x_1, \ldots, x_k)$, with $1 \leq k \leq n$, we shall assume that each x_i is realized on the free port indexed by i. Additionally, in graphical representations we shall always suppose that those labelled with $1, \ldots, k$ are the "rightmost" k free ports, with the labelling increasing from left to right. In other words, the free ports of μ are labelled as follows:



It is then implicit that, for example, the net



realizes $(x_1, \ldots, x_k, x_{k-1})$. With this convention, the interface on which a tuple is realized is implicit in the arity of the tuple, so we shall not need to specify it each time. By the way, in the end we shall be interested in nets realizing tuples whose arity is exactly equal to the number of their free ports, in which case there is no choice for the interface, except for the ordering, which is always assumed to be "from left to right".

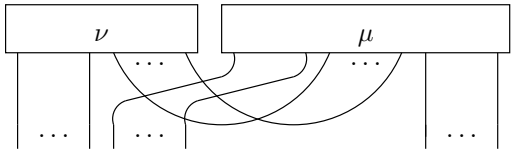
In the following, we shall adopt the uniform notation $(\!|\cdot, \cdot|\!)_\alpha$ for our bijections, where $\alpha \in \{\delta, \zeta\}$, so that $(\!|\cdot, \cdot|\!)_\delta = \langle \cdot, \cdot \rangle$ and $(\!|\cdot, \cdot|\!)_\zeta = [\cdot, \cdot]$. Moreover, we shall use $\vec{x}$ as the generic notation for an element of $\mathcal{C}$.

Definition 3.11 (Realizability relation) *Let μ be a net with at least $n \geq 1$ free ports, and let $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{C}$. We define the relation $\mu \Vdash \vec{x}$, which is read* μ realizes $\vec{x}$*, by induction on $\sharp\vec{x}$:*

- *$\sharp\vec{x} = 0$. In this case, the x_i are all scalars, and the definition is by induction on n:*

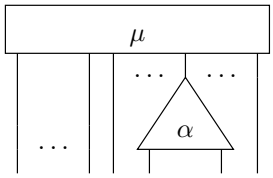
– $n = 1$, *which means that* $\vec{x}$ *is a single scalar* a. *Then,* $\mu \Vdash a$ *iff, whenever* $a \neq \mathbf{0}$, $\mu\Downarrow_1$.

– $n > 1$, *which means that* $\vec{x} = (a_1, \ldots, a_n)$. *Then,* $\mu \Vdash (a_1, \ldots, a_n)$ *iff for all* $0 < i < n$ *and for all* ν *such that* $\nu \Vdash (a_1, \ldots, a_i)$, *the net*



realizes $(a_{i+1}, \ldots, a_n)$.

- $\sharp\vec{x} > 0$. *This means that at least one of the* $x_j i$ *is not a scalar; then,* $\mu \Vdash (x_1, \ldots, (\!|y, z|\!)_\alpha, \ldots, x_n)$ *iff the net*



realizes $(x_1, \ldots, z, y, \ldots, x_n)$ *(of course the port to which the* α *cell is connected is the port corresponding to* $(\!|y, z|\!)_\alpha$*).*

Notice that no condition is required to realize the scalar $\mathbf{0}$: any net realizes it. In particular, $\varepsilon \Vdash \mathbf{0}$; this will be fundamental for the proof of the Adequacy Lemma 3.22.

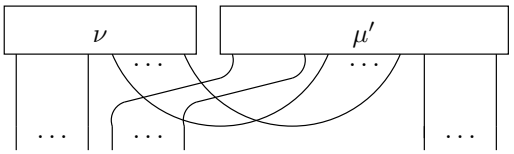
We start by showing a few fundamental properties of the realizability relation.

Lemma 3.13 (Saturation) *Let* μ *be a net, let* $\vec{x} \in \mathcal{C}$, *and let* $\mu \Vdash \vec{x}$. *Then:*

1. $\mu' \to^* \mu$ *implies* $\mu' \Vdash \vec{x}$;
2. $\mu' \simeq_\eta \mu$ *implies* $\mu' \Vdash \vec{x}$.

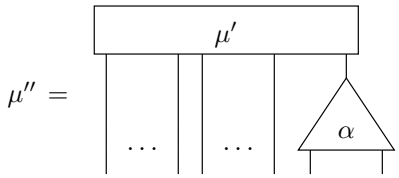
Proof. We start by proving point 1, by induction on $\sharp\vec{x}$. In case $\sharp\vec{x} = 0$, we use a further induction on the arity of $\vec{x}$. In case the arity is 1, the fact that $\mu' \to^* \mu\Downarrow_1$ implies $\mu'\Downarrow_1$ follows trivially from the definition of observable port.

If the arity is strictly greater than 1, we pose $\vec{x} = (x_1, \ldots, x_n)$, and we have to check that, given any $0 < i < n$ and any net ν realizing $(x_1, \ldots, x_i)$, the net

$\mu'' =$ 

realizes $\vec{x}' = (x_{i+1}, \ldots, x_n)$. Now if, inside μ'', we reduce μ' to get μ, we obtain a net which by hypothesis realizes $\vec{x}'$; now, the arity of $\vec{x}'$ is strictly smaller than that of $\vec{x}$, so the induction hypothesis applies, and we have $\mu'' \Vdash \vec{x}'$ for all ν as above. But then we have just proved that $\mu' \Vdash \vec{x}$.

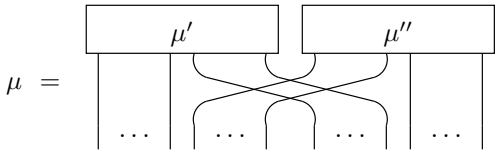
We get to the case in which $\sharp\vec{x} > 0$. We can suppose without loss of generality that $\vec{x} = (\vec{x}', (\!|y, z|\!)_\alpha)$. Then, we want to prove that the net



realizes $\vec{x}'' = (\vec{x}', z, y)$. Now, just as before, we can reduce μ' inside μ'' to get μ, so that by hypothesis we obtain a net realizing $\vec{x}''$. In this case too $\sharp\vec{x}''$ is strictly smaller than $\sharp\vec{x}$, so by induction hypothesis we have that $\mu'' \Vdash \vec{x}''$, which is what we wanted to prove.

Point 2 is proved in exactly the same way; the only difference is the base case, i.e., $\sharp\vec{x} = 0$ and arity 1, in which the argument used is that we know that η-equivalence cannot alter the observability of a port (in fact, $\simeq_\eta \subseteq \simeq$, cf. Proposition 2.15). □

Lemma 3.14 (Mix) *Let $\mu' \Vdash \vec{x}'$ and $\mu'' \Vdash \vec{x}''$. Then, the net*



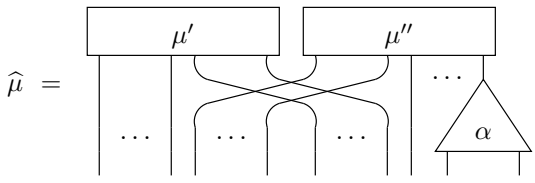
realizes $\vec{x} = (\vec{x}', \vec{x}'')$.

Proof. Again, the proof is by induction on $\sharp\vec{x}$. If $\sharp\vec{x} = 0$, we reason by induction on the arity n of $\vec{x}$. The base case is $n = 2$, which is trivial. If $n > 2$, we need to plug into μ a net ν realizing a "sub-tuple" of $\vec{x}$, and prove that the net thus obtained, which we call $\widehat{\mu}$, realizes "the rest" of $\vec{x}$. There are two cases:

- The first one is that in which $\vec{x}'$ can be decomposed into $(\vec{x}'_1, \vec{x}'_2)$, and $\nu \Vdash x'_1$. Then, we need to prove that $\widehat{\mu} \Vdash (\vec{x}'_2, \vec{x}'')$. But we know by hypothesis that composing ν with μ' yields a net μ_0 realizing $\vec{x}'_2$; now, since $(\vec{x}'_2, \vec{x}'')$ has arity strictly smaller than n, when we juxtapose μ_0 to μ'', we obtain a net which by induction hypothesis realizes $(\vec{x}'_2, \vec{x}'')$; but this net is exactly $\widehat{\mu}$.

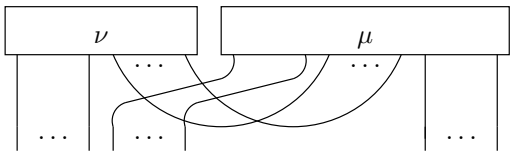
- The second case is that in which $\vec{x}''$ decomposes into $(\vec{x}''_1, \vec{x}''_2)$, and $\nu \Vdash (\vec{x}', \vec{x}''_1)$, so we need to prove that $\widehat{\mu} \Vdash \vec{x}''_2$. Here we do not even need the induction hypothesis: when we plug μ' into ν, we obtain a net μ_0 which realizes $\vec{x}''_1$; now, plugging μ_0 into μ'' yields a net realizing $\vec{x}''_2$, which is exactly $\widehat{\mu}$.

For what concerns the case $\sharp\vec{x} > 0$, we can suppose without loss of generality that $\vec{x}'' = (\vec{x}''_0, (\!|y, z|\!)_\alpha)$, so we need to prove that the net



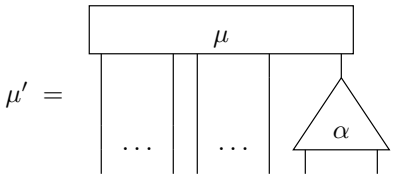
realizes $(\vec{x}', \vec{x}''_0, z, y)$. Now, by hypothesis, if we remove μ' from $\widehat{\mu}$, we obtain a net μ_0 realizing $\vec{y} = (\vec{x}''_0, z, y)$. Of course we have $\sharp\vec{y} < \sharp\vec{x}''$, so the induction hypothesis gives us that the net obtained by juxtaposing μ' and μ_0 realizes $(\vec{x}', \vec{x}''_0, z, y)$; but this net is nothing but $\widehat{\mu}$ itself. □

Lemma 3.15 (Composition) *Let $\mu \Vdash (\vec{x}_1, \vec{x}_2)$. Then, for any net ν such that $\nu \Vdash \vec{x}_1$, the net*



realizes $\vec{x}_2$.

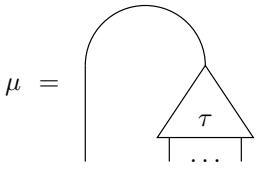
Proof. We pose $\vec{x} = (\vec{x}_1, \vec{x}_2)$, and we reason by induction on $\sharp\vec{x}$. If $\sharp\vec{x} = 0$, then the result follows immediately from the definition. If $\sharp\vec{x} > 0$, we can suppose without loss of generality that $\vec{x}_2 = (\vec{x}'_2, (\!|y, z|\!)_\alpha)$. Now, by definition we know that the net



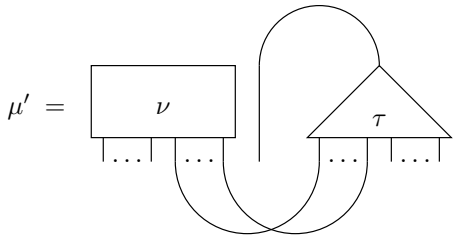
realizes $\vec{x}' = (\vec{x}_1, \vec{x}'_2, z, y)$. Since $\sharp\vec{x}' < \sharp\vec{x}$, the induction hypothesis assures us that when we plug ν into μ', we obtain a net which realizes $(\vec{x}'_2, z, y)$. But then, by definition, removing the α cell from this net yields a net realizing $\vec{x}_2$, which is what we wanted to prove. □

Lemma 3.16 *If W is a single wire, then for all $x \in \Phi_2(\mathcal{A})$, $W \Vdash x$.*

Proof. By induction on $\sharp x$. If x is a scalar, the result a trivial consequence of the definition, since $W{\Downarrow_i}$ obviously holds for any of its free ports i. If x is not a scalar, then its canonical tree with respect to any of the two bijections $(\!|\cdot, \cdot|\!)_\alpha$ (see Definition 3.4) naturally induces a tree τ (in the sense of a net), and, by definition, $\mu \Vdash x$ is equivalent to the fact that the net



realizes $(a_1, \ldots, a_n)$, where n is the arity of τ, and the a_i are the scalars generating x. But then, by definition, proving $\mu \Vdash (a_1, \ldots, a_n)$ amounts to showing that, for any $0 < j < n$ and any net ν realizing $(a_1, \ldots, a_j)$, the net

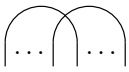


realizes $(a_j + 1, \ldots, a_n)$. We can prove this by induction on n. The base case is $n = 2$ ($n = 1$ would mean that x is a scalar, and $\mu = W$), in which case clearly there is an observable path starting from the "rightmost" port of μ' no matter what net ν we plug into the other leaf of τ. The induction step is trivial: some leaves of τ are connected to ν, leaving a number smaller than n free, to which the induction hypothesis applies. □

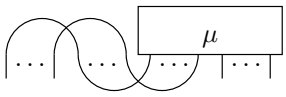
The following is an immediate application of the above results:

Corollary 3.17 *If $\mu \Vdash (\vec{x}_1, \vec{x}_2)$, then $\mu \Vdash \vec{x}_2$.*

Proof. From Lemma 3.16 and the Mix Lemma 3.14, we know that the net



realizes $\vec{x}_1$. Then, by the Composition Lemma 3.15, the net



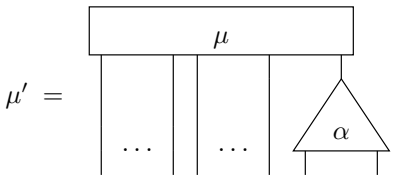
realizes $\vec{x}_2$; but the above net is μ itself! □

It is perhaps useful to observe here the difference between realizing *separately* two elements $x, y \in \Phi_2(\mathcal{A})$ on two free ports, and realizing the couple (x, y) *simultaneously* on the same two ports. For example, as Lemma 3.16 shows, it is correct to write that $W \Vdash x$ on one free port and $W \Vdash y$ on the other; however, as we shall see in Lemma 3.19, $W \Vdash (x, y)$ only if $x = y$.

Lemma 3.18 (Adaptation) *Let $\vec{x} = (x_1, \ldots, x_n) \in \mathcal{C}$, and let $\mu \Vdash \vec{x}$. Then, for all $1 \leq i \leq n$, $x_i \neq \mathbf{0}$ implies $\mu\Downarrow_i$.*

Proof. For convenience, we shall suppose $i = n$. As usual, the proof is by induction on $\sharp x_n$. If x_n is a scalar, by Corollary 3.17 we know that $\mu \Vdash x_n$, and since $x_n \neq \mathbf{0}$, by definition we have $\mu\Downarrow_n$.

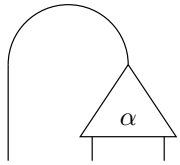
If $x_n = (\!|y, z|\!)_\alpha$, with y not a scalar if $z = \mathbf{0}$, we know by hypothesis and by Corollary 3.17 that the net



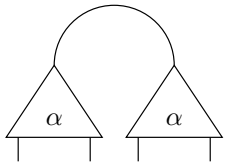
realizes (z, y). Now one of y, z must be different from $\mathbf{0}$; then, by induction hypothesis, we have $\mu'\Downarrow_k$, where k is the free port of μ' associated to the non-zero element. But then part 2 of Lemma 2.3 implies $\mu\Downarrow_n$. □

Lemma 3.19 *Let W be a single wire. Then, for all $x \in \Phi_2(\mathcal{A})$, $W \Vdash (x, x)$.*

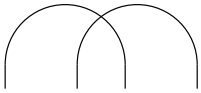
Proof. By induction on $\sharp x$. If x is a scalar, we just need to check that plugging any net ν such that $\nu \Vdash x$ on any extremity of W yields a net realizing x; but this is trivial, since the net obtained in this way is exactly ν itself. If $x = (\!|y, z|\!)_\alpha$, we need to check that the net



realizes $((\!|y, z|\!)_\alpha, z, y)$. But, by definition, this is equivalent to saying that the net

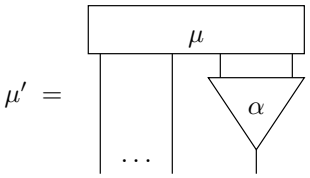


realizes (z, y, z, y). Such a net reduces to



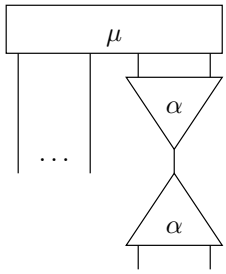
which by induction hypothesis and the Mix Lemma 3.14, does indeed realize (z, y, z, y); we can then conclude by saturation (Lemma 3.13). □

Lemma 3.20 *If $\mu \Vdash (\vec{x}, y, z)$, then, for all $\alpha \in \{\delta, \zeta\}$, the net*



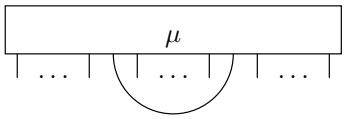
realizes $(\vec{x}, (\!|y, z|\!)_\alpha)$.

Proof. To prove the lemma, it is enough to show that the net



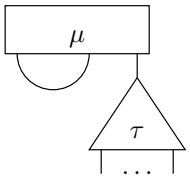
realizes $(\vec{x}, z, y)$. But by hypothesis the above net reduces to a net realizing precisely that tuple, so we conclude by applying the Saturation Lemma 3.13. □

Lemma 3.21 *If $\mu \Vdash (\vec{x}_1, x, \vec{x}_2, x, \vec{x}_3)$, then the net*

$\mu' =$ 

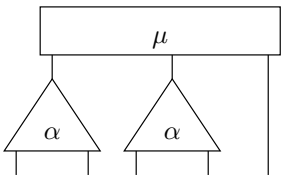
realizes $(\vec{x}_1, \vec{x}_2, \vec{x}_3)$, where the two ports connected by the wire are those corresponding to the two occurrences of x.

Proof. To simplify the notations, we shall ignore $\vec{x}_1$ and $\vec{x}_2$, and we shall assume that $\vec{x}_3$ is of arity 1 and equal to y, so $\mu \Vdash (x, x, y)$; the proof in the general case is identical, but much heavier with respect to the notations. First of all, if τ is the tree naturally induced by the canonical tree of y with respect to any of the two bijections (cf. the proof of Lemma 3.16 and Definition 3.4), then, by definition, saying that $\mu' \Vdash y$ is equivalent to saying that the net

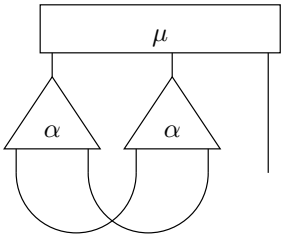


realizes $(a_1, \ldots, a_n)$, where n is the arity of τ and the a_i are the scalars from which y is generated. Therefore, we can assume y to be a scalar; if it is not, we just add the required trees and, as the reader may check, the proof carries on without any difference.

Now we reason by induction on $\sharp x$. Suppose first that x is a scalar. By Lemma 3.19, the wire W realizes (x, x); but notice that μ' can be seen as the result of plugging W into μ, which by hypothesis realizes (x, x, y), so by definition $\mu' \Vdash y$. If $x = (\!|x_1, x_2|\!)_\alpha$, by hypothesis we know that the net



realizes (x_2, x_1, x_2, x_1, y). Therefore, by applying twice the induction hypothesis, we infer that the net



realizes y. But the above net is η-equivalent to μ', so we can conclude thanks to saturation (Lemma 3.13). □

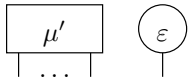
Finally, here comes the crucial *Adequacy Lemma*:

Lemma 3.22 (Adequacy) *Let μ be a net with at least one free port, and let e be an experiment on μ. Then, $\mu \Vdash |e|$.*

Proof. We reason by induction on the number of cells contained in μ. Notice that μ cannot be empty; therefore, if μ contains no cells, it is necessarily a wiring. This implies that $|e|$ is a tuple of even arity, in which elements appear exactly twice. But by Lemma 3.19 and the Mix Lemma 3.14, a wiring always realizes a tuple of this form.

If μ contains at least one cell, we first distinguish the case in which μ has a free principal port. If it is the principal port of an ε cell, then we can assume without loss of generality that

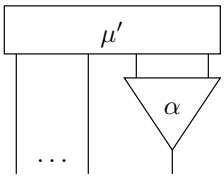
μ =



In this case, $|e| = (\vec{x}, \mathbf{0})$. Now, the restriction e' of e on μ' is clearly such that $|e'| = \vec{x}$, and since μ' has fewer cells than μ, we can use the induction hypothesis and say that $\mu' \Vdash \vec{x}$. Remember now how we observed, right after Definition 3.11, that $\varepsilon \Vdash \mathbf{0}$; hence, an application of the Mix Lemma 3.14 allows us to conclude that $\mu \Vdash (\vec{x}, \mathbf{0})$.

If the free principal port is that of a binary cell α, we can assume that

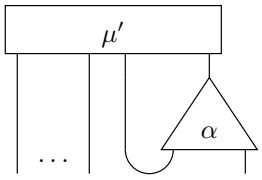
μ =



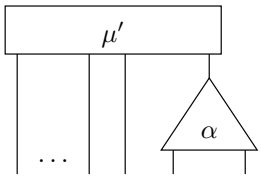
so that $|e| = (\vec{x}, (\!|y, z|\!)_\alpha)$. Again, the restriction e' of e on μ' yields $|e'| = (\vec{x}, y, z)$. Now, μ' being smaller than μ, we can use the induction hypothesis and obtain $\mu' \Vdash (\vec{x}, y, z)$. But then Lemma 3.20 applies, giving us $\mu \Vdash (\vec{x}, (\!|y, z|\!)_\alpha)$.

The only case left is that in which no free port of μ is principal. The interesting configuration is

μ =



In fact, if both auxiliary ports of α are free, the result is trivial consequence of the definition of realizability. Now, in the above situation, we have $|e| = (\vec{x}, x)$. Then, by definition, the restriction of e on μ' must assign the tuple $(\vec{x}, y, (\!|x, y|\!)_\alpha)$ to the free ports of μ', where y is some element of $\Phi_2(\mathcal{A})$. Once again μ' has fewer cells than μ, so an application of the induction hypothesis gives us $\mu' \Vdash (\vec{x}, y, (\!|x, y|\!)_\alpha)$. By definition, this means that the net



realizes $(\vec{x}, y, y, x)$. It is then enough to apply Lemma 3.21 to conclude that $\mu \Vdash (\vec{x}, x)$. □

The following proposition is now an easy corollary of the Adequacy Lemma:

Proposition 3.23 *Let μ be a net with $n \geq 1$ free ports, numbered from 1 to n, and let $(x_1, \ldots, x_n) \in [\![\mu]\!]$. Then, $x_i \neq \mathbf{0}$ implies $\mu\Downarrow_i$.*

Proof. By definition of $[\![\mu]\!]$, the tuple $(x_1, \ldots, x_n)$ must be the result of an experiment on μ, so the Adequacy Lemma tells us that $\mu \Vdash (x_1, \ldots, x_n)$; but then we can conclude by applying Lemma 3.18. □

Corollary 3.24 (Semantical characterization of observability) *Let μ be a net with at least one free port. Then, $\mu\Downarrow$ iff $[\![\mu]\!] \neq \{(\mathbf{0}, \ldots, \mathbf{0})\}$.*

This semantical characterization has as immediate corollary the *adequacy* of the model $\Phi_2(\mathcal{A})$ with respect to the observational equivalence defined in Sect. 2.1:

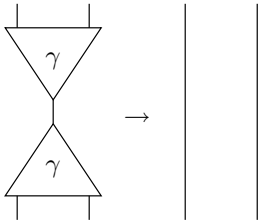
Proposition 3.25 (Semantical adequacy) *For all nets μ, ν, $[\![\mu]\!] = [\![\nu]\!]$ implies $\mu \simeq \nu$, where the interpretation is taken in $\Phi_2(\mathcal{A})$.*

Proof. Consider the contrapositive statement, and assume that there exists a context C such that, for example, $C[\mu]\Downarrow$ and $C[\nu]\Uparrow$. Then, by Corollary 3.24, $[\![C[\nu]]\!] = \{\mathbf{0}\}$, while $[\![C[\mu]]\!]$ contains non-zero elements of $\Phi_2(\mathcal{A})$. But, thanks to Lemma 3.8, $[\![C[\mu]]\!] \neq [\![C[\nu]]\!]$ implies $[\![\mu]\!] \neq [\![\nu]\!]$. □

3.1.6 Why the *symmetric* interaction combinators?

The reader may wonder why we have chosen to work with the *symmetric* combinators instead of the "standard" ones, which enjoy a stronger universality property. The answer is of technical nature: there is a detail in the reduction rules of the interaction combinators which renders impossible the formulation of a relational semantics like the one considered here.

We remind that the "standard" interaction combinators are defined exactly as the symmetric ones, except that instead of ζ there is a binary cell γ, which interacts with itself as follows:

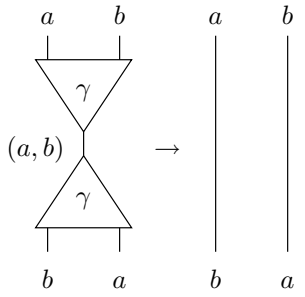


All other interaction rules are unchanged. Notice that the above rule "exchanges" the auxiliary ports of the two γ cells: the first port of each cell is connected to the second port of the other cell. On the contrary, the $\delta\delta$ (and $\zeta\zeta$) annihilation connects first port with first port and second port with second port.

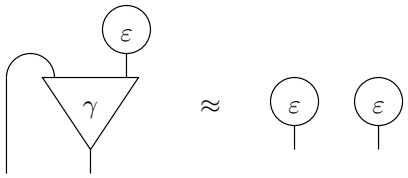
Now, in a relational semantics, reduction is modeled by composition of relations: if the "rightmost" free port of a net μ is connected to the "leftmost" free port of a net μ', the denotational semantics of the resulting net will be (see the proof of Lemma 3.8)

$$[\![\mu]\!] \circ [\![\mu']\!] = \{(a,c)\ ;\ (a,b) \in [\![\mu]\!],\ (b,c) \in [\![\mu']\!]\}.$$

This is ensured by our definition of experiment. But if we try to define experiments in the presence of the $\gamma\gamma$ annihilation, we see that the only way for the interpretation to model reduction is that both of the auxiliary ports of γ cells receive the same value. In fact, in the rule



we clearly need $a = b$. This is an unreasonable restriction; for example, it would imply that the following two nets receive the same semantical interpretation:



These two nets are not $\beta\eta$-equivalent; from Theorem 2.27, we infer that in such a situation, if we ever managed to model $\simeq_\beta$, we would do so by identifying all total nets with a non-empty interface.

The argument given here of course does not rule out the possibility of finding a denotational semantics for the interaction combinators; it simply shows that the standard definitions do not work, and justifies our shift towards the symmetric combinators.

3.2 Full abstraction

The aim of this section is to investigate the possibility of finding a fully-abstract semantics for the symmetric combinators, with respect to the observational equivalence introduced in Sect. 2.1. In other words, we seek a semantics in which $\simeq$ becomes an equality. Such a result would be analogous to Wadsworth

and Hyland's result in the λ-calculus [Wad76, Hyl76], relating the observational equivalence based on head normalization to the local structure of Scott's D_∞ model.

Unfortunately, we do not know at present how to define an interaction set such that its corresponding interpretation satisfies the full abstraction property. Nevertheless, we do have a syntactical model, similar to the model of Bhöm trees in the λ-calculus, which fully corresponds to the observational equivalence. This may serve as a starting point to find a "more semantical" model enjoying full abstraction.

3.2.1 Edifices

We now introduce certain structures, called *edifices*, which can be seen as the symmetric combinators' equivalent of Böhm trees. Because of the symmetries of interaction nets, i.e., the fact that there is no distinguished conclusion, these structures will not be trees (or sets of branches), but rather sets of (unordered) *pairs* of branches, called *arches*, which correspond to observable paths.

Definition 3.12 (Words, biwords, streams) *A* word *is an element of the set $\mathcal{W} = \{\mathbf{p}, \mathbf{q}\}^* \cup \{\mathbf{p}, \mathbf{q}\}^{\mathbb{N}}$, i.e., a finite or infinite binary word.*[2] *The elements of $\mathcal{W}$ will be ranged over by x, y, and the empty word denoted by $\mathbf{1}$.*

A biword *is a pair of words, denoted by $x \otimes y$. A biword $x \otimes y$ is finite iff x and y are finite; the set of finite biwords will be denoted by $\mathcal{S}_{\text{fin}}$, and ranged over by s, t. We define the concatenation of two finite biwords as*

$$(x \otimes y)(x' \otimes y') = xx' \otimes yy'$$

with neutral element $\mathbf{1} \otimes \mathbf{1}$. The length of a finite biword $x \otimes y$ is defined to be the length of x plus the length of y.

A stream *is a biword $x \otimes y$ such that x and y are both infinite. The set of streams is denoted by $\mathcal{S}$, and ranged over by u, v, w. Concatenation can be extended in the obvious way into an operation from $\mathcal{S}_{\text{fin}} \times \mathcal{S}$ to $\mathcal{S}$. We say that a finite biword s is a prefix of a stream u when $u = su'$ for some stream u'. In this case, we write $s < u$.*

In the following, we fix a denumerably infinite set $\mathcal{L}$ of *locations*. The most natural choice is to take $\mathcal{L} = \mathbb{N}$, but no algebraic structure is needed on locations. The elements of $\mathcal{L}$ will be ranged over by i, j. A *base* is a finite set of locations; bases will be ranged over by I.

Definition 3.13 (Pillar) *A* pillar *is an element of $\mathcal{S} \times \mathcal{L}$, denoted by $u@i$, and ranged over by ξ, υ. The pillar $u@i$ is said to be* based at *i.*

Definition 3.14 (Arch) *An* arch *is an unordered pair of pillars, denoted by $\xi \frown \upsilon$, and ranged over by $\mathfrak{a}$. Unordered means that the expressions $\xi \frown \upsilon$ and $\upsilon \frown \xi$ denote the same arch. An arch is* based at *the locations at which its pillars are based.*

Definition 3.15 (Edifice) *An* edifice *of base I is a set $\mathfrak{E}$ of arches based at elements of I.*

[2] The notations have been chosen to be reminiscent of those used in the geometry of interaction (cf. Sect. 3.3)

Given an edifice $\mathfrak{E}$ of base I, and given $i, j \in I$, we can consider the set of arches of $\mathfrak{E}$ based at i, j:

$$\mathcal{A}_{i,j}(\mathfrak{E}) = \{\mathfrak{a} \in \mathfrak{E} \mid \mathfrak{a} = u@i \frown v@j\}.$$

Notice that, by definition, $\mathcal{A}_{j,i}(\mathfrak{E}) = \mathcal{A}_{i,j}(\mathfrak{E})$.[3]

The set $\mathcal{A}_{i,j}(\mathfrak{E})$ can be turned into a topological space. First of all, we apply the usual Cantor distance on streams:

$$d(u, v) = 2^{-q}$$

where q is the length of the longest common prefix of u, v. Then, we define the metric

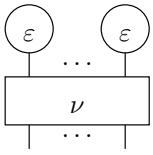
$$D(u@i \frown v@j, u'@i \frown v'@j) = \max\{d(u, u'), d(v, v')\}.$$

Clearly, the set of streams being a Cantor space, the topology we obtain is homeomorphic to the product topology on $\mathcal{S} \times \mathcal{S}$ (which is still a Cantor space).

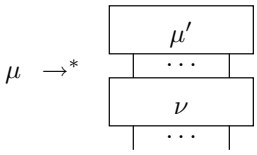
Definition 3.16 (Completion) *An edifice $\mathfrak{E}$ of base I is said to be* complete *iff, for all $i, j \in I$, $\mathcal{A}_{i,j}(\mathfrak{E})$ is a complete metric space (with respect to the metric define above). The* completion *of an edifice $\mathfrak{E}$ of base I, denoted by $\overline{\mathfrak{E}}$, is the union of the completions of all of the spaces $\mathcal{A}_{i,j}(\mathfrak{E})$ for $i, j \in I$, i.e., the smallest complete edifice containing $\mathfrak{E}$.*

3.2.2 Edifices as infinite cut-free forms

Definition 3.17 (Approximation) *Let μ_0, μ be two nets with the same number of free ports. We say that μ_0 approximates μ, or that μ_0 is an* approximation *of μ, and we write $\mu_0 \sqsubseteq \mu$, iff μ_0 is a cut-free net of the form*



and



for some net μ'.

Observe that, if μ is a net with n free ports, then the net with n free ports consisting only of ε combinators is an approximation of μ: simply take, in the picture of Definition 3.17, ν to be the identity wiring and $\mu' = \mu$. In fact, from the semantical point of view, we have that $\mu_0 \sqsubseteq \mu$ implies $[\![\mu_0]\!] \subseteq [\![\mu]\!]$, where the interpretation is taken in any interaction set. To see this, observe that,

[3] This corresponds to Proposition 3.38 in the geometry of interaction, i.e., the fact that the interpretation of a net is "hermitian".

by definition, $\mu_0 \sqsubseteq \mu$ implies that μ reduces to a net having μ_0 as a cut-free subnet, modulo some ε cells. With the notations of Definition 3.17, we can write $\mu_0 = \nu[\varepsilon]$ and $\mu \to^* \nu[\mu']$, where ε denotes a net made of the suitable number of ε cells. Now, given an experiment on μ_0, it is always possible to turn it into an experiment of $\nu[\mu']$: simply assign $\mathbf{0}$ to all ports of μ'. Therefore, $[\![\mu_0]\!] \subseteq [\![\nu[\mu']]\!]$. But the interpretation is stable under reduction, hence the claim.

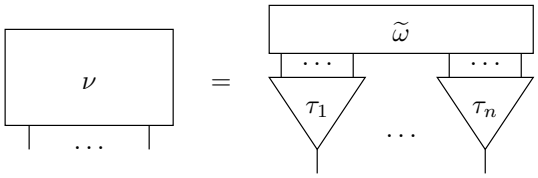
Of course, if μ is total with cut-free form ν, then $\nu \sqsubseteq \mu$, and ν is "maximal" among the approximations of μ. If μ is not total, then there is no "maximal" cut-free net approximating it. What is needed is a sort of infinite cut-free form, i.e., a cut-free form which exists even if a net is not total, just as Böhm trees can be seen as infinite normal forms of λ-terms which exist even if they are non-normalizable (or unsolvable). This inspires the following definitions.

Definition 3.18 (Address of a leaf) *We associate an element of $\mathcal{S}_{\mathrm{fin}}$ to each leaf of a tree τ, called its address in τ, by induction on τ:*[4]

- *if $\tau = \mathbf{1}$, we associate to its only leaf the empty word $\mathbf{1}$;*
- *if $\tau = \delta(\tau_1, \tau_2)$, and the leaf in question is a leaf of τ_1 (resp. τ_2) whose address in that tree is s, then its address in τ is $(\mathbf{p} \otimes \mathbf{1})s$ (resp. $(\mathbf{q} \otimes \mathbf{1})s$);*
- *if $\tau = \zeta(\tau_1, \tau_2)$, and the leaf in question is a leaf of τ_1 (resp. τ_2) whose address in that tree is s, then its address in τ is $(\mathbf{1} \otimes \mathbf{p})s$ (resp. $(\mathbf{1} \otimes \mathbf{q})s$).*

In the sequel, we shall always assume that the free ports of a net are labelled by locations; for example, we can assume that the free ports of a net μ with n free ports are labelled by the integers in the base $I = \{1, \ldots, n\}$. For this reason, we shall say that a net has base I iff I is as above and μ has n free ports. Moreover, we shall assume that all nets have non-empty interfaces, so that I is never empty.

Definition 3.19 (Edifice of a cut-free net) *Let ν be a reduced net of base I. Remember that ν can always be decomposed as*



We define the edifice of ν, which will be denoted by $\mathfrak{E}^\bullet(\nu)$ and whose base will be I, as the set such that, for all $w \in \mathcal{S}$, $sw@i \frown tw@j \in \mathfrak{E}^\bullet(\nu)$ iff there exist a leaf of τ_i whose address is s and a leaf of τ_j whose address is t which are connected by a wire of $\widetilde{\omega}$.

Notice that the edifice of a cut-free net is either empty (when the net consists solely of ε-packages) or infinite, and in that case its cardinality is $\aleph_1$. Additionally, it has the property of being complete:

Lemma 3.26 *Let ν be a cut-free net of base I. Then, $\mathfrak{E}^\bullet(\nu)$ is complete.*

[4] This address is nothing but the GoI weight of the path starting from the leaf and going down to the root of τ, see Definition 3.23.

PROOF. Let $i, j \in I$, let $\mathfrak{a}_1, \ldots, \mathfrak{a}_n, \ldots$ be a Cauchy sequence of $\mathcal{A}_{i,j}(\mathfrak{E}^\bullet(\nu))$, and let the q be maximum of the lengths of all addresses (which we remind are finite biwords) of the leaves of ν. Because the sequence considered is Cauchy, we know that there exists $k \geq 1$ such that, if we choose $m, n \geq k$, we have $D(\mathfrak{a}_m, \mathfrak{a}_n) < 2^{-q}$. Now put $\mathfrak{a}_m = u_m@i \frown v_m@j$ and $\mathfrak{a}_n = u_n@i \frown v_n@j$, and suppose that the longest common prefixes between u_m, u_n and v_m, v_n have resp. lengths p, r; we have

$$D(\mathfrak{a}_m, \mathfrak{a}_n) = \max\{d(u_m, u_n), d(v_m, v_n)\} = \max\{2^{-p}, 2^{-r}\} < 2^{-q},$$

so $p, r > q$. Now, by Definition 3.19, the streams u_m and u_n must extend two addresses s_m and s_n of ν. But u_m and u_n agree on a prefix which is strictly longer than any such address, so they actually extend the same address; this also true for v_m and v_n. Therefore, using again Definition 3.19, there exist in ν two leaves of resp. addresses s and t such that, for all $n \geq k$, we have $\mathfrak{a}_n = sw_n@i \frown tw_n@j$ for some stream w_n. The limit of the sequence is obviously of the form $su@i \frown tv@j$; but if $u \neq v$, then, for n big enough, there would exist an $\mathfrak{a}_n$ of a form different from the one given above. Hence, the limit of the sequence must also have the form $sw@i \frown tw@j$, and is thus in $\mathfrak{E}^\bullet(\nu)$ by definition. □

Definition 3.20 (Approximate edifices) *Let μ be a net of base I. We define the set of* approximate edifices *of μ as*

$$\mathfrak{Apx}(\mu) = \{\mathfrak{E}^\bullet(\mu_0); \mu_0 \sqsubseteq \mu\}.$$

Definition 3.21 (Edifice of a net) *Let μ be a net of base I. The edifice of μ, denoted by $\mathfrak{E}(\mu)$ and of base I, is the completion of the lub of $\mathfrak{Apx}(\mu)$, i.e.,*

$$\mathfrak{E}(\mu) = \overline{\bigcup \mathfrak{Apx}(\mu)}.$$

We remark that the above definition is compatible with Definition 3.19. In fact, if ν is cut-free, it is easy to see that, for all $\nu_0 \sqsubseteq \nu$, we have $\mathfrak{E}^\bullet(\nu_0) \subseteq \mathfrak{E}^\bullet(\nu)$, which implies $\bigcup \mathfrak{Apx}(\nu) = \mathfrak{E}^\bullet(\nu)$. Therefore, by Lemma 3.26, we obtain $\mathfrak{E}(\nu) = \mathfrak{E}^\bullet(\nu)$.

3.2.3 Edifices as semantics

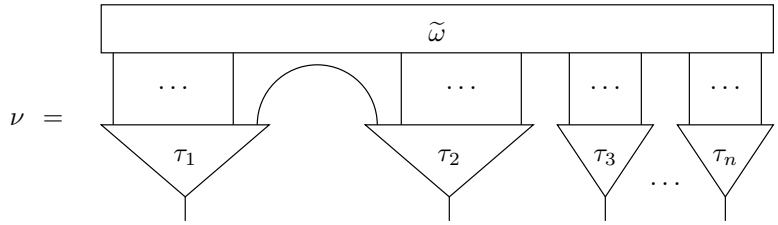
The interpretation of nets as edifices actually gives an alternative denotational semantics for the symmetric combinators. In fact, we have the following:

Lemma 3.27 *Let $\mu \to^* \mu'$. Then, $\mathfrak{E}(\mu) = \mathfrak{E}(\mu')$.*

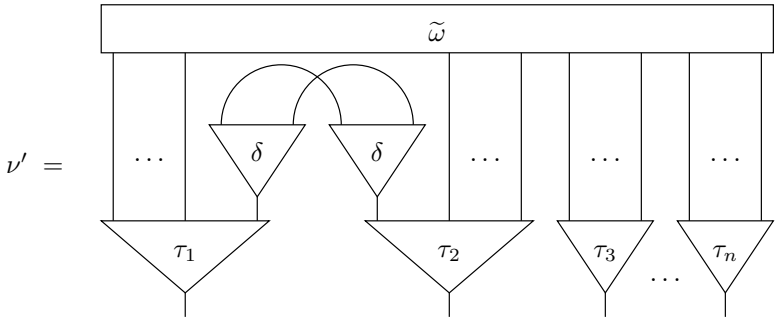
PROOF. By definition of edifice of a net, it is enough to prove that $\mathfrak{Apx}(\mu) = \mathfrak{Apx}(\mu')$. Suppose $\mu_0 \sqsubseteq \mu$. By definition, we have $\mu \to^* \nu[\mu_1]$, where ν is cut-free and $\mu_0 = \nu[\varepsilon]$, where ε represents a suitable net consisting solely of ε cells. By confluence, $\nu[\mu_1] \to^* \mu'$; but ν is cut-free and there are no active pairs between ν and μ_1, so we must have $\mu' = \nu[\mu_1']$, where $\mu_1 \to^* \mu_1'$, which means that μ_0 is an approximation of μ' too. The converse is even more trivial: whatever reduct of μ' is a reduct of μ as well. □

Lemma 3.28 *Let ν, ν' be two cut-free nets such that $\nu \simeq_\eta \nu'$. Then, $\mathfrak{E}^\bullet(\nu) = \mathfrak{E}^\bullet(\nu')$.*

PROOF. It is easy to see that the addresses of leaves are invariant under the $\delta\zeta$ equation. The case of an equation involving ε is trivial, because ε cells produce no arch in the edifice of a cut-free net. It remains to check the $\delta\delta$ and $\zeta\zeta$ equations. Firs of all observe that, since ν and ν' are cut-free, these equations can only be applied to a wire connecting two auxiliary ports, or an auxiliary port and a free port, or two free ports; in fact, if this were not the case, there would be active pairs in one of ν, ν'. Let us concentrate on the $\delta\delta$ equation, the $\zeta\zeta$ equation being structurally identical. We can suppose without loss of generality that



and



(τ_1 may be equal to τ_2, and the wire in ν may then connect two leaves of the same tree, but this does not pose any problem to the argument we shall give below). Suppose that s and t are the addresses of the two leaves of resp. τ_1 and τ_2 connected by the wire shown in ν. Then, the four "new" leaves of ν' receive the following addresses, from left to right in the above picture: $s(\mathbf{p}\otimes\mathbf{1})$, $s(\mathbf{q}\otimes\mathbf{1})$, $t(\mathbf{p}\otimes\mathbf{1})$, and $t(\mathbf{q}\otimes\mathbf{1})$. This means that $s(\mathbf{p}\otimes\mathbf{1})w@1 \frown t(\mathbf{p}\otimes\mathbf{1})w@2 \in \mathfrak{E}^\bullet(\nu')$ and $s(\mathbf{q}\otimes\mathbf{1})w@1 \frown t(\mathbf{q}\otimes\mathbf{1})w@2 \in \mathfrak{E}^\bullet(\nu')$ for all w; clearly, these arches belong to $\mathfrak{E}^\bullet(\nu)$ as well, because, by definition, $\mathfrak{E}^\bullet(\nu)$ contains the arches $sw@1 \frown tw@2$ for all w. But all arches of this latter form are also in $\mathfrak{E}^\bullet(\nu')$; in fact, if we pose $w = x \otimes y$, since x is infinite, it must either be of form $\mathbf{p}x'$ or $\mathbf{q}x'$, so that either $w = (\mathbf{p}\otimes\mathbf{1})(x'\otimes y)$ or $w = (\mathbf{q}\otimes\mathbf{1})(x'\otimes y)$. □

Proposition 3.29 *$\mu \simeq_{\beta\eta} \nu$ implies $\mathfrak{E}(\mu) = \mathfrak{E}(\nu)$.*

PROOF. Define $\mu_0 \sqsubseteq_{\beta\eta} \mu$ iff $\mu_0 \simeq_{\beta\eta} \mu_0'$, $\mu \simeq_{\beta\eta} \mu'$ and $\mu_0' \sqsubseteq \mu'$. Then, by Lemmas 3.27 and 3.28 we have that the set

$$\{\mathfrak{E}^\bullet(\mu_0); \mu_0 \sqsubseteq_{\beta\eta} \mu\}$$

coincides with $\mathfrak{Apx}(\mu)$, hence the thesis. □

There is also a characterization of observability which is similar to that given by Corollary 3.24, which this time comes straight-forwardly from the definitions:

Proposition 3.30 *A net μ is observable iff $\mathfrak{E}(\mu) \neq \emptyset$.*

Proof. Suppose $\mu \Downarrow$. Then, by definition, there is an approximation of μ different from the net containing only ε cells, hence $\mathfrak{E}(\mu) \neq \emptyset$. Conversely, if $\mu \Uparrow$, then by definition no observable path ever develops between any free port of μ, so $\mathfrak{E}(\mu) = \emptyset$. □

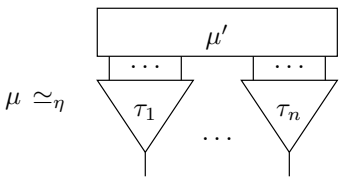
The following result is fundamental. It is the reason why metric completeness has been considered, instead of a simple union of successive approximations.

Lemma 3.31 *Let μ, μ' be two nets of base I. Then, $\mathfrak{E}(\mu) \neq \mathfrak{E}(\mu')$ implies that there exist $i, j \in I$, two finite biwords s, t, and two cut-free nets $\mu_0 \sqsubseteq \mu$ and $\mu'_0 \sqsubseteq \mu'$ such that, if we put $\mathfrak{a}_w = sw@i \frown tw@j$, either for all w $\mathfrak{a}_w \in \mathfrak{E}^\bullet(\mu_0) \setminus \mathfrak{E}(\mu')$, or for all w $\mathfrak{a}_w \in \mathfrak{E}^\bullet(\mu'_0) \setminus \mathfrak{E}(\mu)$.*

Proof. Set $\mathfrak{E}_0 = \bigcup \mathfrak{Apx}(\mu)$. Suppose, without loss of generality, that there exists $\mathfrak{a} \in \mathfrak{E}(\mu) \setminus \mathfrak{E}(\mu')$, based at $i, j \in I$. If such an arch has been added in completing $\mathfrak{E}_0$, then there exists a Cauchy sequence $\mathfrak{a}_n \in \mathcal{A}_{i,j}(\mathfrak{E}_0)$ of limit $\mathfrak{a}$. Since a subsequence of a Cauchy sequence is still a Cauchy sequence, there must exists an integer m such that, for all $n \geq m$, $\mathfrak{a}_n \in \mathfrak{E}_0 \setminus \mathfrak{E}(\mu')$, otherwise $\mathfrak{a}$ would belong to $\mathfrak{E}(\mu')$ because of its completeness. Therefore, modulo replacing it by one of these $\mathfrak{a}_n$, we can always assume that $\mathfrak{a} \in \mathfrak{E}_0$, i.e., that it has not been added in the completion process. If it is so, then by definition there exists an approximation μ_0 of μ such that $\mathfrak{a} \in \mathfrak{E}^\bullet(\mu_0)$, which means that $\mathfrak{a} = sw_0@i \frown tw_0@j$ and, for every stream w, $sw@i \frown tw@j \in \mathfrak{E}^\bullet(\mu_0)$, where s and t are the addresses of two leaves of μ_0 connected by a wire. Now let $s'_1, \ldots, s'_n, \ldots$ be a sequence of prefixes of increasing length of w_0, and set, for all n, $s_n = ss'_n$ and $t_n = ts'_n$. Suppose that, for all n, there exist two streams u_n, v_n such that $\mathfrak{a}_n = s_n u_n@i \frown t_n v_n@j \in \mathfrak{E}(\mu')$; it is not hard to verify that the arches $\mathfrak{a}_n$ would form a Cauchy sequence of limit $\mathfrak{a}$, and thus, by the completeness of $\mathfrak{E}(\mu')$, we would obtain $\mathfrak{a} \in \mathfrak{E}(\mu')$, a contradiction. Therefore, there must exist an integer n such that, for all w, $s_n w@i \frown t_n w@j \in \mathfrak{E}^\bullet(\mu_0) \setminus \mathfrak{E}(\mu')$. □

The next result is similar to the Equivalence Lemma 2.24, even though here we do not have any requirement concerning totality:

Lemma 3.32 *Let μ be a net with n free ports, and let $\tau_1, \ldots, \tau_n$ be trees. Then, there exists a net μ' such that*

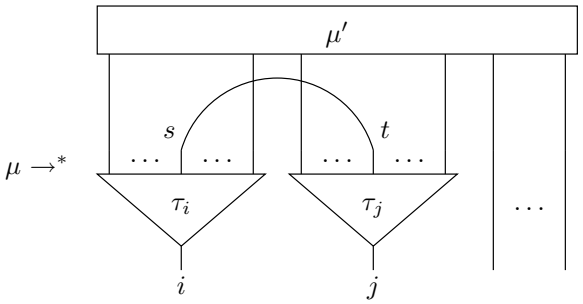


Proof. A trivial application of Lemma 2.21. □

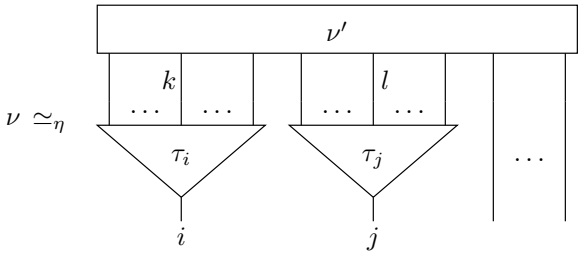
We can now show that edifices completely characterize the observational equivalence $\simeq$:

Theorem 3.33 (Full abstraction) *Let μ, ν be two nets with $n \geq 1$ free ports. Then, $\mathfrak{E}(\mu) = \mathfrak{E}(\nu)$ iff $\mu \simeq \nu$.*

PROOF. We shall only show the actual full abstraction property, i.e., that $\mu \simeq \nu$ implies $\mathfrak{E}(\mu) = \mathfrak{E}(\nu)$; the other implication is a consequence of Proposition 3.30. For this, we consider the contrapositive statement, and assume $\mathfrak{E}(\mu) \neq \mathfrak{E}(\nu)$. Let I be the base of μ and ν, and let $i, j \in I$. We shall assume $i \neq j$; the reader is invited to check that the argument can be straight-forwardly adapted to the case $i = j$. Set $\mathfrak{E} = \mathfrak{E}(\mu)$ and $\mathfrak{F} = \mathfrak{E}(\nu)$. By Lemma 3.31, we know that there exists $\mu_0 \sqsubseteq \mu$ and two leaves of μ_0 of addresses s, t such that, for all w, $sw@i \frown tw@j \in \mathfrak{E}^\bullet(\mu_0) \setminus \mathfrak{F}$ (it could actually be that these arches belong to $\mathfrak{E}^\bullet(\nu_0) \setminus \mathfrak{E}$, where $\nu_0 \sqsubseteq \nu$, but obviously our assumtion causes no loss of generality). By Definition 3.19, and by the fact that $\mu_0 \sqsubseteq \mu$, we have

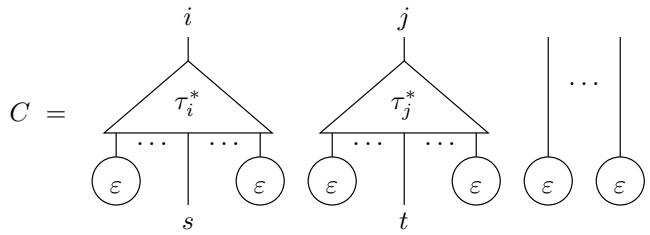


where we have explicitly drawn the connection between the two leaves of resp. addresses s and t. On the other hand, by Lemma 3.32, we have

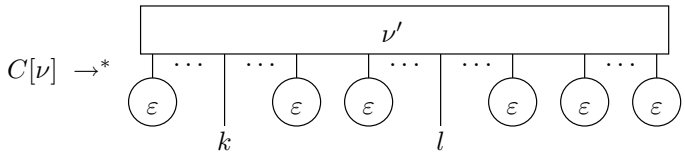


where we have called k and l the two free ports of ν' corresponding resp. to the addresses t and s in of τ_i and τ_j. Observe that, by Proposition 3.29, the edifice of the net on the right is still $\mathfrak{F}$. Now if, in any reduct of ν', there appeared an observable path between k and l, then we would contradict the fact that, for all w, $sw@i \frown tw@j \notin \mathfrak{F}$. Therefore, no observable path ever develops between k and l, which means that ν' is *relatively blind* on $\{k, l\}$ (see Sect. 2.2.3).

Consider then the context

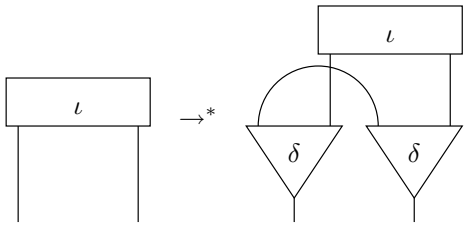


where we have used the canonical antitrees defined at p. 74, and we have left free only the leaves corresponding to the addresses s and t of τ_i and τ_j. Now clearly $C[\mu]\Downarrow$; on the other hand, we have

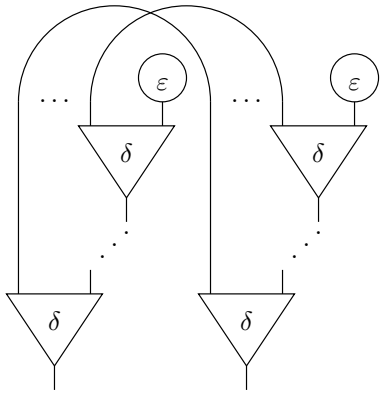


But ν' is relatively blind on $\{k, l\}$, so by Claim 2.3.1 (Sect. 2.2.3), $C[\nu]\Uparrow$. □

As an immediate application of Theorem 3.33, we give an example of a net which is not total, and yet is observationally equivalent to a wire; this is analogous to Wadsworth's "infinitely η-expanding" term $J = RR$, where $R = \lambda xzy.z(xxy)$, which is well known to be observationally equivalent to the identity $I = \lambda z.z$, if one considers the observational equivalence based on solvability (and not on normalizability). In fact, consider a net with the following property:



Such a net exists by the arguments given in Sect. 1.3.2, p. 40. Clearly, the approximations of ι are all and only of the form



Observe that all approximations of ι are "almost" η-equivalent to a wire: they just lack the "rightmost" connection. This connection forms only in the limit, and this is what the completion accounts for. In fact, if we call 1 and 2 the two free ports of ι, in $\bigcup \mathfrak{Apx}(\iota)$ there is, for all n and for every word y, an arch of the form

$$\mathfrak{a}_n = (\mathbf{q}^n \mathbf{p} x \otimes y)@1 \frown (\mathbf{q}^n \mathbf{p} x \otimes y)@2,$$

where x is an infinite word. The arches $\mathfrak{a}_n$ form a Cauchy sequence converging to $(\mathbf{q}^\infty \otimes y)@1 \frown (\mathbf{q}^\infty \otimes y)@2$, where $\mathbf{q}^\infty$ is the word consisting of an infinite sequence of $\mathbf{q}$'s. But then we have

$$\mathfrak{E}(\iota) = \{u@1 \frown u@2 \ ; \text{ for every stream } u\},$$

which is exactly $\mathfrak{E}(W) = \mathfrak{E}^\bullet(W)$, where W is the net consisting of a single wire. Hence, by Theorem 3.33, we have $\iota \simeq W$.

3.3 The Geometry of Interaction

In this section we develop an algebraic semantics for the symmetric combinators, in the style of Girard's Geometry of Interaction (GoI) [Gir89], already sketched by Lafont [Laf97]. We do so in connection with the denotational semantics introduced above, and prove that there is a strong link between the two.

3.3.1 Interaction monoids

Up to here, we have seen that any infinite pointed set $\mathcal{D}$ can serve as the domain for the denotational semantics of nets of symmetric combinators. We shall see that it may be of interest to add some algebraic structure to $\mathcal{D}$; the least we can require is that $\mathcal{D}$ is a monoid.

In the following, the composition $u \circ v$ of two monoid endomorphisms is denoted simply uv.

Definition 3.22 (Interaction monoid) *An* interaction monoid *is a commutative monoid $(M, +, 0)$ admitting eight endomorphisms $\mathbf{c}, \mathbf{c}^*, \mathbf{d}, \mathbf{d}^*, \mathbf{f}, \mathbf{f}^*, \mathbf{g}, \mathbf{g}^*$ such that the functions $\langle x, y\rangle = \mathbf{c}(x) + \mathbf{d}(y)$ and $[x, y] = \mathbf{f}(x) + \mathbf{g}(y)$ are companion isomorphisms between $M \oplus M$ and M, and $\mathbf{c}^*, \mathbf{d}^*$ and $\mathbf{f}^*, \mathbf{g}^*$ are their respective projections.*

Proposition 3.34 *A commutative monoid $(M, +, 0)$ is an interaction monoid iff there exist eight endomorphisms $\mathbf{c}, \mathbf{c}^*, \mathbf{d}, \mathbf{d}^*, \mathbf{f}, \mathbf{f}^*, \mathbf{g}, \mathbf{g}^*$ of M such that:*

1. $\mathbf{c}^*\mathbf{c} = \mathbf{d}^*\mathbf{d} = \mathbf{f}^*\mathbf{f} = \mathbf{g}^*\mathbf{g} = \mathbf{1}$, *where* $\mathbf{1}$ *is the identity on* M;
2. $\mathbf{c}^*\mathbf{d} = \mathbf{d}^*\mathbf{c} = \mathbf{f}^*\mathbf{g} = \mathbf{g}^*\mathbf{f} = \mathbf{0}$, *where* $\mathbf{0}$ *is the everywhere-zero endomorphism on* M;
3. $\mathbf{c}\mathbf{c}^* + \mathbf{d}\mathbf{d}^* = \mathbf{f}\mathbf{f}^* + \mathbf{g}\mathbf{g}^* = \mathbf{1}$;
4. $\mathbf{c}, \mathbf{c}^*, \mathbf{d}, \mathbf{d}^*$ *commute with* $\mathbf{f}, \mathbf{f}^*, \mathbf{g}, \mathbf{g}^*$.

Proof. Let us first prove that Definition 3.22 implies the four statements above:

1. By definition, for every $x \in M$, $\mathbf{c}(x) = \langle x, 0\rangle$, and by the hypothesis that $\mathbf{c}^*$ is the left projection of $\langle\cdot,\cdot\rangle$, we obtain $\mathbf{c}^*\mathbf{c}(x) = \mathbf{c}^*(\langle x, 0\rangle) = x$. The same applies to the other annihilations.
2. As above, for every $x \in M$ we have $\mathbf{d}(x) = \langle 0, x\rangle$, from which we obtain $\mathbf{c}^*\mathbf{d}(x) = \mathbf{c}^*(\langle 0, x\rangle) = 0$. The same applies to the other annihilations.
3. From the surjectivity of $\langle\cdot,\cdot\rangle$, given $x \in M$ we know that there exist $y, z \in M$ such that $x = \langle y, z\rangle$. Then, we have
$$(\mathbf{c}\mathbf{c}^* + \mathbf{d}\mathbf{d}^*)(x) = \mathbf{c}\mathbf{c}^*(x) + \mathbf{d}\mathbf{d}^*(x) = \mathbf{c}(y) + \mathbf{d}(z) = \langle y, z\rangle = x.$$
The case $\mathbf{f}\mathbf{f}^* + \mathbf{g}\mathbf{g}^* = \mathbf{1}$ is identical.
4. To prove that $\mathbf{c}, \mathbf{d}$ commute with $\mathbf{f}, \mathbf{g}$ simply consider that, by the companion hypothesis, for all $x \in M$, $\langle [x, 0], 0\rangle = [\langle x, 0\rangle, 0]$, from which we get $\mathbf{c}\mathbf{f}(x) = \mathbf{f}\mathbf{c}(x)$, and $\langle [0, x], 0\rangle = [0, \langle x, 0\rangle]$, from which we obtain $\mathbf{c}\mathbf{g}(x) = \mathbf{g}\mathbf{c}(x)$, and so on.

To prove that $\mathbf{c}, \mathbf{d}$ commute to $\mathbf{f}^*, \mathbf{g}^*$ consider, given a generic $x \in M$, the (unique) decomposition $x = [y, z]$, so that

$$\mathbf{f}^*\mathbf{c}(x) = \mathbf{f}^*\mathbf{c}\mathbf{f}(y) + \mathbf{f}^*\mathbf{c}\mathbf{g}(z) = \mathbf{f}^*\mathbf{f}\mathbf{c}(y) + \mathbf{f}^*\mathbf{g}\mathbf{c}(z) = \mathbf{c}(y) = \mathbf{c}\mathbf{f}^*(x),$$

where we have used point 1 and 2 proved above. The other cases are handled similarly, as also the commutations between $\mathbf{f}, \mathbf{g}$ and $\mathbf{c}^*, \mathbf{d}^*$.

To prove that $\mathbf{c}^*, \mathbf{d}^*$ commute to $\mathbf{f}^*, \mathbf{g}^*$, we consider the same decomposition above for the generic $x \in M$, and we obtain

$$\mathbf{f}^*\mathbf{c}^*(x) = \mathbf{f}^*\mathbf{c}^*\mathbf{f}(y) + \mathbf{f}^*\mathbf{c}^*\mathbf{g}(z) = \mathbf{f}^*\mathbf{f}\mathbf{c}^*(y) + \mathbf{f}^*\mathbf{g}\mathbf{c}^*(z) = \mathbf{c}^*(y) = \mathbf{c}^*\mathbf{f}^*(x),$$

and similarly for the other cases.

Assume now that the eight endomorphisms verify the four statements above. The fact that the maps $(x, y) \mapsto \mathbf{c}(x) + \mathbf{d}(y)$ and $(x, y) \mapsto \mathbf{f}(x) + \mathbf{g}(y)$ are homomorphisms from $M \oplus M$ to M is obvious; we need to prove that they are bijective. We shall do it for the first map, the second being structurally identical.

Suppose that, given $x, x', y, y' \in M$, $\mathbf{c}(x) + \mathbf{d}(y) = \mathbf{c}(x') + \mathbf{d}(y')$; then, applying $\mathbf{c}^*$ (resp. $\mathbf{d}^*$) to both sides of the equation and using points 1 and 2, we get $x = x'$ (resp. $y = y'$), which proves injectivity. For what concerns surjectivity, by point 3 for any element $x \in M$ there exist $y, z \in M$ such that $x = \mathbf{c}(y) + \mathbf{d}(z)$: just pose $y = \mathbf{c}^*(x)$ and $z = \mathbf{d}^*(x)$. The fact that $\mathbf{c}^*, \mathbf{d}^*$ are the projections associated to this isomorphism is trivial.

We are left to proving that the two isomorphisms are companions. If we pose $\langle x, y\rangle = \mathbf{c}(x) + \mathbf{d}(y)$ and $[x, y] = \mathbf{f}(x) + \mathbf{g}(y)$, using point 4 we have

$$\begin{aligned}\langle [w, x], [y, z]\rangle = \; & \mathbf{c}\mathbf{f}(w) + \mathbf{c}\mathbf{g}(x) + \mathbf{d}\mathbf{f}(y) + \mathbf{d}\mathbf{g}(z) = \\ & \mathbf{f}\mathbf{c}(w) + \mathbf{f}\mathbf{d}(y) + \mathbf{g}\mathbf{c}(x) + \mathbf{g}\mathbf{d}(z) = [\langle w, y\rangle, \langle x, z\rangle],\end{aligned}$$

which completes the proof. □

If $(A, +, 0)$ is a commutative monoid, then the set $\Phi_2(A)$ of all almost everywhere-zero sequences of elements of A indexed by pairs of non-negative integers is an example of interaction monoid. Addition is defined pointwise, and the neutral element is the everywhere-zero sequence; the eight endomorphisms are defined as follows:

$$\mathbf{c}(x)_{m,n} = \begin{cases} x_{k,n} & \text{if } m = 2k \\ 0 & \text{if } m = 2k+1 \end{cases} \qquad \mathbf{c}^*(x)_{m,n} = x_{2m,n}$$

$$\mathbf{d}(x)_{m,n} = \begin{cases} 0 & \text{if } m = 2k \\ x_{k,n} & \text{if } m = 2k+1 \end{cases} \qquad \mathbf{d}^*(x)_{m,n} = x_{2m+1,n}$$

$$\mathbf{f}(x)_{m,n} = \begin{cases} x_{m,k} & \text{if } n = 2k \\ 0 & \text{if } n = 2k+1 \end{cases} \qquad \mathbf{f}^*(x)_{m,n} = x_{m,2n}$$

$$\mathbf{g}(x)_{m,n} = \begin{cases} 0 & \text{if } n = 2k \\ x_{m,k} & \text{if } n = 2k+1 \end{cases} \qquad \mathbf{g}^*(x)_{m,n} = x_{m,2n+1}$$

from which it is not hard to check that points 1, 2, and 3 of Proposition 3.34 are satisfied. For what concerns point 4, just notice that $\mathbf{c}, \mathbf{c}^*, \mathbf{d}, \mathbf{d}^*$ and $\mathbf{f}, \mathbf{f}^*, \mathbf{g}, \mathbf{g}^*$ act on separate indexes, so all operations commute.

Interaction monoids are clearly interaction sets (the distinguished element is the zero of the monoid), so Definitions 3.7 and 3.8 can be applied just as they are, yielding a semantics that interprets a net with $n \geq 1$ free ports as a subset of M^n, where $M^n = M \oplus \cdots \oplus M$. In particular, the interpretation using an interaction monoid of the form $\Phi_2(A)$ as above obviously has all the properties proved in Sect. 3.2, most notably it is fully abstract with respect to $\simeq$.

Since we are considering monoids, it makes sense to add experiments pointwise, i.e., given a net μ and two experiments e_1, e_2 on μ over an interaction monoid M, we can define the function $e_1 + e_2$ from $\mathsf{Ports}(\mu)$ to M that associates to a port i the element $e_1(i) + e_2(i)$. One may wonder whether this yields another experiment; the answer is indeed positive:

Lemma 3.35 (Additivity) *Let μ be a net, and e_1, e_2 two experiments on μ over an interaction monoid M. Then, $e_1 + e_2$ is an experiment.*

Proof. The fact that $e_1 + e_2$ respects conditions (a) and (d) of Definition 3.7 is obvious. Conditions (b) and (c) are consequences of the fact that our companion bijections are monoid isomorphisms. For example, in the case of a δ cell, whose auxiliary and principal ports are resp. i, j, and k, we have

$$\begin{aligned}(e_1 + e_2)(k) &= e_1(k) + e_2(k) = \langle e_1(i), e_1(j)\rangle + \langle e_2(i), e_2(j)\rangle = \\ &= \langle e_1(i) + e_2(i), e_1(j) + e_2(j)\rangle = \langle (e_1 + e_2)(i), (e_1 + e_2)(j)\rangle.\end{aligned}$$

The case of a ζ cell is identical. □

Therefore, if μ is a net with $n \geq 1$ free ports, $[\![\mu]\!]$ is not just any subset of M^n, it is a *submonoid*:

Corollary 3.36 *Let μ be a net with $n \geq 1$ free ports, and M an interaction monoid. Then, the interpretation of μ in M is a submonoid of M^n.*

Proof. By the Additivity Lemma 3.35, the only thing left to verify is that $0 \in [\![\mu]\!]$, which is obvious. □

3.3.2 The GoI semantics

Given an interaction monoid M, we shall now define a semantics which interprets a cut-free net with $n \geq 1$ free ports as an endomorphism of $M^n = M \oplus \cdots \oplus M$. This is just a reformulation of what already done by Lafont [Laf97], so most proofs will be omitted. Our only original contribution is the proof of Theorem 3.43.

In the following, we denote by R the sub-semiring (with unit) of $\mathrm{End}(M)$ generated by $\mathbf{c}, \mathbf{c}^*, \mathbf{d}, \mathbf{d}^*, \mathbf{f}, \mathbf{f}^*, \mathbf{g}, \mathbf{g}^*$.

Definition 3.23 (Weight) *Let M be an interaction monoid, μ a net, and p a straight path of μ (see Definition 2.1). We define the* weight *of p in M, which is an element of R and is denoted $w(p)$, by induction on the length of p:*

- *p contains just one port: $w(p) = \mathbf{1}$ (the identity endomorphism);*
- *$p = p' \cdot i$, and the ending port of p' and i do not belong to the same cell: $w(p) = w(p')$.*

– $p = p' \cdot i$, *where* p' *ends with the left (resp. right) auxiliary port of a* δ *cell, and* i *is the principal port of the same* δ *cell:* $w(p) = \mathbf{c}w(p')$ *(resp.* $w(p) = \mathbf{d}w(p')$*);*

– $p = p' \cdot i$, *where* p' *ends with the principal port of a* δ *cell, and* i *is the left (resp. right) auxiliary port of the same* δ *cell:* $w(p) = \mathbf{c}^* w(p')$ *(resp.* $w(p) = \mathbf{d}^* w(p')$*);*

– $p = p' \cdot i$, *where* p' *ends with the left (resp. right) auxiliary port of a* ζ *cell, and* i *is the principal port of the same* ζ *cell:* $w(p) = \mathbf{f}w(p')$ *(resp.* $w(p) = \mathbf{g}w(p')$*);*

– $p = p' \cdot i$, *where* p' *ends with the principal port of a* ζ *cell, and* i *is the left (resp. right) auxiliary port of the same* ζ *cell:* $w(p) = \mathbf{f}^* w(p')$ *(resp.* $w(p) = \mathbf{g}^* w(p')$*).*

Given a graph-theoretical path, one can always consider its *reversal*, i.e., the same path walked from target to source. Notice that the reversal of a straight path is still straight. The unit semiring R can be equipped with an involution $(\cdot)^*$:

- $(\mathbf{c})^* = \mathbf{c}^*$, $(\mathbf{c}^*)^* = \mathbf{c}$, and similarly for the other generators;
- $\mathbf{0}^* = \mathbf{0}$, and for all $u, v \in W$, $(u + v)^* = u^* + v^*$;
- $\mathbf{1}^* = \mathbf{1}$, and for all $u, v \in W$, $(uv)^* = v^* u^*$.

It is then straight-forward to check the following:

Lemma 3.37 (Reversal) *Let* μ *be a net,* p *a straight path of* μ*, and* p' *the reversal of* p*. Then,* $w(p') = w(p)^*$*.*

By definition, if p is a straight path, $w(p)$ is either the identity, or a composition of the endomorphisms $\mathbf{c}, \mathbf{d}, \mathbf{g}, \mathbf{f}$ and $\mathbf{c}^*, \mathbf{d}^*, \mathbf{g}^*, \mathbf{f}^*$. We call such compositions (plus the identity as well) *monomials*. It is easy to see that, if $w(p) \neq \mathbf{0}$, then $w(p)$ is equal to a monomial of the form ab^*, where a is a composition of $\mathbf{c}, \mathbf{d}, \mathbf{g}, \mathbf{f}$ and b^* is a composition of $\mathbf{c}^*, \mathbf{d}^*, \mathbf{g}^*, \mathbf{f}^*$.

We are now ready to define the GoI interpretation of a cut-free net:

Definition 3.24 (GoI interpretation) *Let* M *be an interaction monoid, let* ν *be a cut-free net with* $n \geq 1$ *free ports, and let* P_{ji} *be the set of straight paths of* ν *starting from the free port* j *and ending into the free port* i*. The GoI interpretation of* ν *in* M *is an endomorphism of* M^n*, which we represent as a formal* $n \times n$ *matrix* $\nu^\bullet$*, whose entries are defined as follows:*

$$\nu^\bullet_{ij} = \sum_{p \in P_{ji}} w(p).$$

i and j range over the free ports of ν, and the sum is intended to be equal to $\mathbf{0}$ *(the everywhere-zero endomorphism) if* $P_{ji} = \emptyset$*.*

If A is a formal matrix with coefficients in R, we can define A^* as the transpose-involute matrix of A: $(A^*)_{ij} = (A_{ji})^*$. Then, the following clearly holds from Lemma 3.37 applied to Definition 3.24:

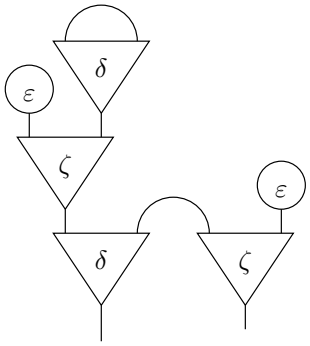
Proposition 3.38 *If $\nu^\bullet$ is the GoI interpretation of a cut-free net ν, then*

$$\nu^{\bullet *} = \nu^\bullet.$$

We can give a few examples to clarify the definition. If ε, δ, and W are the three nets defined in Sect. 3.1.2, p. 108, we have $\varepsilon^\bullet = \mathbf{0}$, $\delta^\bullet = \mathbf{cd}^* + \mathbf{dc}^*$, while $W^\bullet$ is the "flip" endomorphism of $M \oplus M$, i.e., $W^\bullet(x \oplus y) = y \oplus x$. It is represented by the following matrix:

$$W^\bullet = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} \end{bmatrix}.$$

A slightly more complicated example is the net

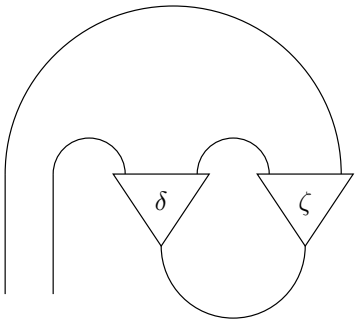


whose GoI interpretation is

$$\begin{bmatrix} \mathbf{cg}(\mathbf{cd}^* + \mathbf{dc}^*)\mathbf{g}^*\mathbf{c}^* & \mathbf{df}^* \\ \mathbf{fd}^* & \mathbf{0} \end{bmatrix}.$$

In all cases, the reader can check that Proposition 3.38 is verified.

The reader may wonder why we have restricted our interpretation to cut-free nets, in sharp contrast to Definition 3.8, where the denotational semantics is defined for *any* net. The reason is quite simple: in the absence of any restriction, Definition 3.24 would not make sense in general, since P_{ji} may contain an infinite number of non-zero-weighing paths. As a matter of fact, consider the following example:

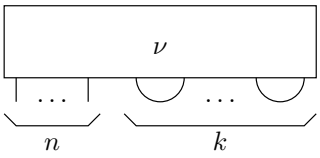


There is obviously an infinite number of straight paths from, for instance, the left free port to the right one; their weights are $\mathbf{c}^*(\mathbf{fd}^*)^n\mathbf{g}$, for every $n \in \mathbb{N}$. No element of R can be associated to the sum of all these paths, so the interpretation would be undefined. On the other hand, the following result assures us that Definition 3.24 is sound as we formulated it:

Proposition 3.39 *Let ν be a cut-free net with at least one free port, and let i, j be two free ports (maybe the same) of ν. Then, P_{ji} is finite.*

PROOF. Remember the general decomposition of a cut-free net with $n \geq 1$ free ports: n trees $\tau_1, \ldots, \tau_n$ with an ε-wiring $\widetilde{\omega}$ "on top". Now, a straight path p from port j to port i is necessarily of the following shape: p goes up along one branch of τ_j from the root to its leaf, which is connected through a wire of $\widetilde{\omega}$ to a leaf k of τ_i; p follows this connection, and then goes down the branch of τ_i leading from k to its root. Therefore, the number of straight paths in P_{ji} is bounded by the number of leaves of τ_j, which is of course finite. □

Any net with $n \geq 1$ free ports and k active pairs and/or vicious circles can be decomposed as follows:



where ν is cut-free and has $n+2k$ free ports. Notice that, because of the possible presence of vicious circles, ν is not unique in general. Nevertheless, given an interaction monoid M and a net μ with at least one free port and k active pairs and/or vicious circles, we can associate to μ at least one endomorphism $\mu^\bullet$ of M^{n+2k}, which is the GoI interpretation of the net ν in one of the possible decompositions; the association will be unique exactly when μ does not contain vicious circles.

We also consider the endomorphism $\sigma_{n,k}$ of M^{n+2k} defined by the formal matrix

$$\sigma_{n,k} = \begin{bmatrix} \mathbf{0} & & & & & & & \\ & \ddots & & & & & & \\ & & \mathbf{0} & & & & & \\ & & & \mathbf{0} & \mathbf{1} & & & \\ & & & \mathbf{1} & \mathbf{0} & & & \\ & & & & & \ddots & & \\ & & & & & & \mathbf{0} & \mathbf{1} \\ & & & & & & \mathbf{1} & \mathbf{0} \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \} n \\ \\ \\ \\ \\ \} 2k \\ \\ \end{matrix}$$

(the entries not specified are $\mathbf{0}$) and the homomorphism $\pi_{n,k}$, which is the inclusion of M^n into M^{n+2k}:

$$\pi_{n,k} = \begin{bmatrix} \mathbf{1} & & \\ & \ddots & \\ & & \mathbf{1} \\ \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \begin{matrix} \\ \\ \\ \\ \\ \\ \end{matrix} \begin{matrix} \} n \\ \\ \\ \\ \} 2k \\ \\ \end{matrix}$$

We just write σ and π when n and k are clear from the context, or when we do not want to specify them.

We can now state the main theorem of the GoI semantics, which is the interaction combinators' equivalent of Girard's fundamental result for linear logic [Gir89]:

Proposition 3.40 (Lafont [Laf97]) *Let μ be a total net with $n \geq 1$ free ports and k active pairs and/or vicious circles, and let $\mu^\bullet$ be the (unique) endomorphism associated to μ in an interaction monoid M. Then, if ν is the cut-free form of μ, we have that its GoI interpretation is given by Girard's* execution formula

$$\nu^\bullet = \mathrm{Ex}(\mu^\bullet, \sigma) = \pi^t \left(\sum_{i=0}^{\infty} \mu^\bullet (\sigma \mu^\bullet)^i \right) \pi,$$

where π^t is the transpose of π.

The execution formula makes sense because $\sigma\mu^\bullet$ is nilpotent. This is a consequence of the fact that the execution formula is an invariant of reduction. Therefore, if μ is total (which means that $\mu^\bullet$ is uniquely defined), then $\mathrm{Ex}(\mu^\bullet, \sigma)$ can be seen as a semantics for μ. In particular, if μ is cut-free, $\sigma = \mathbf{0}$ and $\mathrm{Ex}(\mu^\bullet, \mathbf{0}) = \mu^\bullet$.

Proposition 3.40 tells us in particular that if a net μ is total, then $\sigma\mu^\bullet$ is nilpotent. In the rest of the section, we shall prove that the converse holds as well. The result is analogous to that found by Danos&Regnier for the strongly normalizable terms of the λ-calculus [DR95].

Definition 3.25 (Regular path, Danos-Regnier) *A straight path p is* regular *iff $w(p) \neq \mathbf{0}$.*

Recall now maximal paths as introduced in Definition 2.17; here, we shall prove a result similar to Proposition 2.17 considering only *regular* maximal paths.

Definition 3.26 (Regularly well-founded net) *A generic net μ is* regularly well-founded *iff for each cell α of μ, there is a finite non-null number of maximal regular paths starting from α (see p. 81, just before Definition 2.18, for the meaning of "starting from α").*

Lemma 3.41 *Let $\mu \to^* \mu'$. Then, μ is regularly well-founded iff μ' is.*

Proof. By Proposition 3.40, regular paths are invariant under reduction. □

Lemma 3.42 *If a net is regularly well-founded, then it is total.*

Proof. Even though a regularly well-founded net is not necessarily well-founded, the proof carries over exactly as that of Proposition 2.17, using Lemma 3.41. □

Theorem 3.43 (Characterization of totality by nilpotency) *Let μ be a net, and let $\mu^\bullet$ be the GoI interpretation of μ in an interaction monoid M. Then, μ is total iff $\sigma\mu^\bullet$ is nilpotent.*

Proof. We need only prove the "if" part, the "only if" being already given by Proposition 3.40. By Lemma 3.42, it is enough to prove that if $\sigma\mu^\bullet$ is nilpotent, then μ is regularly well-founded.

First of all, referring to the decomposition of a generic net given at p. 139, we can partition the interface of ν into two parts: one corresponding to the free ports of μ, which we shall call *visible interface*, the other corresponding to the active pairs and vicious circles of μ, which we shall call *hidden interface*. As a consequence, we can use a block representation for $\mu^\bullet$ (which we remind is defined to be equal to $\nu^\bullet$), and write

$$\mu^\bullet = \begin{bmatrix} A & B^* \\ B & C \end{bmatrix}$$

where A, B, C are suitable formal matrices: A contains the weights of the paths from the visible interface to itself, C contains the weights of the paths from the hidden interface to itself, and B, B^* contain the weights of the paths between the two interfaces. Of course, A and C satisfy $A^* = A$ and $C^* = C$. Similarly, we put

$$\sigma = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_1 \end{bmatrix}$$

where σ_1 is the only non-zero block of σ (cf. p. 139).

Now consider a cell α of μ (and hence of ν). We shall analyze how straight paths starting from α may develop, paying particular attention to those that are regular and maximal. Let us call τ the tree containing α. A maximal path starting from α must first of all descend to the root of τ; this path, which we call ϕ, is in fact the prefix of any possible maximal path starting from α. There are now two possibilities: either the root of τ is in the visible interface, or it is in the hidden interface. In the first case, ϕ is maximal (and regular), and thus cannot be extended further. In the second case, the path may continue by following a wire to get to the root of another tree belonging to the hidden interface. From the root of this tree the path continues inside ν, until reaching either an ε cell, or a port of the visible interface, or a port of the hidden interface. In the first two cases, we have a maximal path; in the second, we start all over again. We can thus call n-path a straight path starting from α obtained after applying n times the above process: the only 0-path is that we called ϕ, while 1-paths are those obtained extending ϕ by passing once again through ν, and so on.

We now claim the following:

Claim 3.43.1 *For all n, there is only a finite number of n-paths.*

Proof. We can prove this by induction: in case $n = 0$, we have already said that there is only one 0-path, ϕ; for what concerns $n + 1$-paths, these are built by taking an n-path and extending it through a free-port-to-free-port straight path of ν. But ν is cut-free, so Proposition 3.39 assures us that n-paths can be extended in only finitely many ways; we apply the induction hypothesis, and we are done. □

The construction of n-paths can be described algebraically by applying the matrices $\mu^\bullet$ and σ to a vector representing the initial path ϕ. In fact, $\mu^\bullet$ is nothing but the adjacency matrix of the (multi)graph whose vertices are the free ports of ν, and whose (weighed) edges are the straight paths between them. Whenever there is at least one path between two free ports, we see a sum of monomials appearing in the corresponding entry of $\mu^\bullet$; when no path exists, we see $\mathbf{0}$. On the other hand, σ is nothing but the adjacency matrix of the

graph representing the connections between the free ports of ν accounting for the active pairs and vicious circles of μ; these are the edges (of weight $\mathbf{1}$) which are used each time we restart the process described above, to build an $n+1$-path out of an n-path.

So suppose that $w(\phi) = u$, and put

$$\upsilon = \begin{bmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ u \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

where the number of elements of the vector υ is equal to the number of free ports of ν, u being placed in the position corresponding to the root of τ. In general, we can split υ into two components U, V as follows:

$$\upsilon = \begin{bmatrix} V \\ U \end{bmatrix}$$

V corresponds to the visible interface, and U to the hidden interface. Then, we have that all of the weights of all n-paths can be obtained with the following product:

$$(\mu^{\bullet}\sigma)^n \upsilon = \left(\begin{bmatrix} A & B^* \\ B & C \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \sigma_1 \end{bmatrix} \right)^n \begin{bmatrix} V \\ U \end{bmatrix}$$

Now, for all n, the result of the above product is of the form

$$\begin{bmatrix} V_n \\ U_n \end{bmatrix}$$

where $V_0 = V$ and $U_0 = U$. The elements of V_n are sums of monomials, each monomial accounting for a finite number of maximal regular n-paths. The fact that these paths are regular is obvious (if they were not, their weight would not appear as a monomial!), while their maximality is justified by considering that the paths whose weight appears in V_n are those starting from α and ending into the visible interface of ν, which is the interface, i.e., the free ports, of μ. The fact that the correspondence monomial/path is one to many must be ascribed to η-expansion: there may be two paths from α to a free port, one weighing $\mathbf{cc}^*$ and the other weighing $\mathbf{dd}^*$, whose sum is just the monomial $\mathbf{1}$.

Not all maximal regular n-paths are taken into account though: those ending into ε cells in fact do not appear in V_n. These however are only a finite number: in fact, a path ending into an ε cell must be the extension of an n-path for some n; but Claim 3.43.1 states that there are only finitely many of these.

Therefore, there is a correspondence between the number of monomials appearing in V_n and the number of n-paths. If we denote by $|V_n|$ the number of monomials appearing in V_n, and by p_n the number of n-paths, we have in fact that

$$p_n \leq k_n |V_n| + c_n,$$

where k_n and c_n are suitable non-negative integers, the first accounting for the one-to-many correspondence monomials/paths, and the second accounting for those n-paths ending into ε cells. Notice that c_n can be at most equal to the number of non-maximal regular $n-1$-paths, i.e., those ending into a port of the hidden interface of ν. But these paths appear in U_{n-1}; therefore, we can further extend our inequality and write

$$p_n \leq k_n|V_n| + k'_n|U_{n-1}| \leq K_n(|V_n| + |U_{n-1}|),$$

where k'_n is another suitable non-negative integer, and K_n is the maximum between k_n and k'_n; conventionally, we fix $|V_{-1}| = |U_{-1}| = 0$.

Now we can bound the total number of maximal regular paths starting from α, which we denote by T, as follows:

$$T = \sum_{i=0}^{\infty} p_i \leq \sum_{i=0}^{\infty} K_i(|V_i| + |U_{i-1}|) \leq \sum_{i=0}^{\infty} K_i(|(\mu^\bullet\sigma)^i\upsilon| + |(\mu^\bullet\sigma)^{i-1}\upsilon|),$$

where we have used the obvious notation $|(\mu^\bullet\sigma)^i\upsilon| = |V_i| + |U_i|$. But by hypothesis $\sigma\mu^\bullet$ is nilpotent, which means that $\mu^\bullet\sigma$ is nilpotent too; then, the above sum is finite.

We have thus shown that, under the hypothesis that $\sigma\mu^\bullet$ is nilpotent, there are only finitely many maximal regular paths starting from any cell α of μ. To complete the proof, we need to show that there is at least one. This is easy if we look at the contrapositive statement, i.e., we prove that the non-existence of maximal regular paths for some cell α of μ implies that $\sigma\mu^\bullet$ is not nilpotent.

So suppose that there is a cell α of μ having no maximal regular path starting from it, and consider again the construction of n-paths described above. We start from α, and build the 0-path ϕ, which is regular, of weight u. The port into which ϕ ends cannot be in the visible interface of ν, otherwise ϕ would be maximal; as a consequence, we can extend ϕ, and build several 1-paths. We claim that at least one of these is regular:

Claim 3.43.2 *For all n, there exists at least one regular n-path.*

Proof. By induction on n. If $n = 0$, the claim holds since ϕ is regular. So suppose we have built a regular n-path ϕ', whose weight must be a monomial of the form ab^*. ϕ' must end into a port of the hidden interface of ν (otherwise it would be maximal), so we can take a wire and keep going from the root of some tree still rooted at the hidden interface of ν. Now we can use the weight of ϕ' to know where to go:

- suppose we are at the principal port of a δ cell. If the weight is $\mathbf{c}u'$ (resp. $\mathbf{d}u'$), we go up through its left (resp. right) auxiliary port, and keep going with the weight u'. Otherwise, if the weight is ab^* and a contains no $\mathbf{c}$ or $\mathbf{d}$, then we are free to choose any auxiliary port, and the weight becomes $a\mathbf{c}^*b^*$ or $a\mathbf{d}^*b^*$, depending on our choice;
- suppose we are at the principal port of a ζ cell. We do the same as above, with $\mathbf{f}, \mathbf{g}, \mathbf{f}^*, \mathbf{g}^*$ replacing resp. $\mathbf{c}, \mathbf{d}, \mathbf{c}^*, \mathbf{d}^*$.

After this, we cannot arrive to an ε cell, otherwise we would contradict our hypothesis; therefore, we arrive at some leaf of some tree of of ν, connected to

some other tree (maybe the same). From here, we descend down a tree to its root, thus obtaining a non-null weight v and arriving at a port of the hidden interface of ν, where we can start over. □

Now the weight of the path whose existence is claimed above appears in $(\mu^\bullet\sigma)^n v$, which means that $\mu^\bullet\sigma$ is not nilpotent. □

3.3.3 Relationship between denotational semantics and GoI

We have already seen (Corollary 3.36) that, given a net μ with $n \geq 1$ free ports, the denotational semantics of μ in an interaction monoid M is a submonoid of M^n. In case μ is total, and if its GoI semantics in M is $\mu^\bullet$, we shall see that $[\![\mu]\!]$ is the submonoid of the fixpoints of $\mathrm{Ex}(\mu^\bullet, \sigma)$.

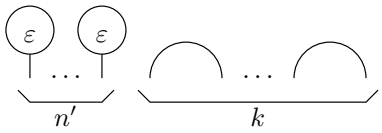
We prove the result stated above for cut-free nets only; by the preservation of both $[\![\cdot]\!]$ and $\mathrm{Ex}(\cdot,\cdot)$ under reduction, this is enough for the result to hold in the more general case of total nets. In the following, we write $\mathrm{fix}(u)$ for the set of fixpoints of a function u.

Theorem 3.44 *Let ν be a cut-free net with $n \geq 1$ free ports. Then*

$$[\![\nu]\!] = \mathrm{fix}(\nu^\bullet),$$

where the interpretations are taken in any interaction monoid M.

Proof. We prove both inclusions be induction on the number m of binary cells of ν. If $m = 0$, then ν is an ε-wiring, containing n' ε cells and k wires, with $n' + 2k = n$. We can assume without loss of generality that ν has the following shape

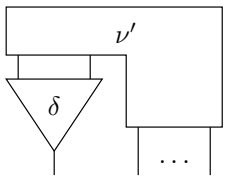


hence $\nu^\bullet = \sigma_{n',k}$. Now, if $x \in [\![\nu]\!]$, then

$$x = (\underbrace{0, \ldots, 0}_{n'}, x_1, x_1, \ldots, x_k, x_k),$$

so that we obviously have $\nu^\bullet(x) = x$. Conversely, it is trivial to check that every fixpoint of $\sigma_{n',k}$ is of this form.

Now let $m > 0$; then, since ν is cut-free, at least one of its free ports is the principal port of a binary cell. Suppose it is a δ cell; we can assume without loss of generality that ν has the following shape



where ν' is also cut-free. Now, if we order from left to right the free ports of ν and ν', we have that their GoI interpretations are given by the following formal matrices:

$$\nu'^{\bullet} = \left[\begin{array}{cc|c} h & u^* & A^* \\ u & k & B^* \\ \hline A & B & C \end{array}\right]$$

$$\nu^{\bullet} = \left[\begin{array}{c|c} \mathbf{c}h\mathbf{c}^* + \mathbf{d}u^*\mathbf{c}^* + \mathbf{c}u\mathbf{d}^* + \mathbf{d}k\mathbf{d}^* & \mathbf{c}A^* + \mathbf{d}B^* \\ \hline A\mathbf{c}^* + B\mathbf{d}^* & C \end{array}\right].$$

We have used the block notation to represent an arbitrary number (even zero) of entries in the bottom-right part of the matrices, corresponding to the $n-1$ ports that are free both in ν and ν'. Here, h, u, and k are endomorphisms of R, with $h^* = h$ and $k^* = k$, while A, B and C are resp. $(n-1) \times 1$ and $(n-1) \times (n-1)$ matrices with entries in R, with $C^* = C$.

Let us now take $x \in [\![\nu]\!]$. We know that x is an element of M^n, so $x = (y, z)$, where $y \in M$ and $z \in M^{n-1}$. Moreover, since y is associated to the free ports of a δ cell, we have $y = \mathbf{c}(y') + \mathbf{d}(y'')$ for some $y', y'' \in M$ such that $x' = (y', y'', z) \in [\![\nu']\!]$. If we apply $\nu'^{\bullet}$ to x', we get

$$\left[\begin{array}{cc|c} h & u^* & A^* \\ u & k & B^* \\ \hline A & B & C \end{array}\right] \cdot \left[\begin{array}{c} y' \\ y'' \\ \hline z \end{array}\right] = \left[\begin{array}{c} h(y') + u^*(y'') + A^*(z) \\ u(y') + k(y'') + B^*(z) \\ \hline A(y') + B(y'') + C(z) \end{array}\right].$$

But by induction hypothesis, $\nu'^{\bullet}(x') = x'$, so the following equalities hold:

$$\begin{aligned} h(y') + u^*(y'') + A^*(z) &= y' \\ u(y') + k(y'') + B^*(z) &= y'' \\ A(y') + B(y'') + C(z) &= z\ . \end{aligned}$$

From this, if we compute $\nu^{\bullet}(x)$, we get

$$\left[\begin{array}{c|c} \mathbf{c}h\mathbf{c}^* + \mathbf{d}u^*\mathbf{c}^* + \mathbf{c}u\mathbf{d}^* + \mathbf{d}k\mathbf{d}^* & \mathbf{c}A^* + \mathbf{d}B^* \\ \hline A\mathbf{c}^* + B\mathbf{d}^* & C \end{array}\right] \cdot \left[\begin{array}{c} \mathbf{c}(y') + \mathbf{d}(y'') \\ \hline z \end{array}\right] =$$

$$= \left[\begin{array}{c} \mathbf{c}(h(y') + u^*(y'') + A^*(z)) + \mathbf{d}(u(y') + k(y'') + B^*(z)) \\ \hline A(y') + B(y'') + C(z) \end{array}\right] = x.$$

Consider now a fixpoint x of $\nu^{\bullet}$. Again, this is an element of M^n, and can be decomposed into (y, z) with $y \in M$ and $z \in M^{n-1}$. Now, by surjectivity there exist $y', y'' \in M$ such that $y = \mathbf{c}(y') + \mathbf{d}(y'')$, so we can define $x' = (y', y'', z)$, for which the same computations done above show that $\nu'^{\bullet}(x') = x'$. By induction hypothesis, $x' \in [\![\nu']\!]$, which means that there is an experiment of ν' with result (y', y'', z); the "same" experiment then gives (y, z) on ν, which proves that $x \in [\![\nu]\!]$.

The proof in the case of a ζ combinator is identical: we just need to replace $\mathbf{c}, \mathbf{d}$ with $\mathbf{f}, \mathbf{g}$. □

Theorem 3.44 tells us that, for any cut-free net ν with at least one free port, if we know $\nu^{\bullet}$, we also know $[\![\nu]\!]$. Is the converse true? The rest of the section is devoted to prove that it is actually the case.

Let M be an interaction monoid, and let ν be a cut-free net with $n \geq 1$ free ports. By Theorem 3.11, we know that the elements of $[\![\nu]\!]$ are described by a balanced bracket expression B. From this expression, it is not hard to build an endomorphism ϕ of M^n such that $\mathrm{fix}(\phi) = [\![\nu]\!]$:

- Suppose that $\mathsf{var}(\mathsf{B}) = \{x_1, \ldots, x_m\}$. We make a new expression B' (no longer balanced) which is structurally identical to B, but such that, for each variable x_i of B, one occurrence of x_i is replaced by x_i' and one by x_i'', with x_i', x_i'' distinct and fresh, and each occurrence of $\mathsf{0}$ is replaced by a distinct fresh variable z_j; this is always possible since B is balanced. Notice that B' contains every variable at most once.

- From the fact that $\langle\cdot,\cdot\rangle$ and $[\cdot,\cdot]$ are bijections, we know that, for each $d \in M^n$, there exist unique $d_1', d_1'', \ldots, d_m', d_m'', e_1, \ldots, e_k \in M$ such that $d = \mathsf{B}'\{\ldots\ x_i' := d_i', x_i'' := d_i''\ \ldots\ z_j := e_j\ \ldots\}$ (see Sect. 3.1.4, p. 113). We then define ϕ as follows:

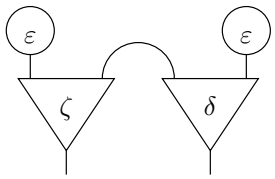
$$\phi(d) = \mathsf{B}'\{\ldots\ x_i' := d_i'', x_i'' := d_i'\ \ldots\ z_j := 0\ \ldots\},$$

  i.e., we "swap" the elements assigned to x_i' and x_i'', and we set each z_j to 0.

- Clearly, $\phi(0) = 0$, and because $\langle\cdot,\cdot\rangle, [\cdot,\cdot]$ are isomorphisms, we also have $\phi(x + y) = \phi(x) + \phi(y)$, so ϕ is indeed an endomorphism of M^n (not an isomorphism though, since in general some non-zero elements may be mapped to zero). It is not hard to check that ϕ verifies $\phi^3 = \phi$, i.e., it is a *partial symmetry*.

- By construction, the fixpoints of ϕ are those elements described by B, so $\mathrm{fix}(\phi) = \mathrm{Im}(\mathsf{B})$.

Let us look at an example to clarify the construction above. The balanced expression $([\mathsf{0}, x], \langle x, \mathsf{0}\rangle)$, which generates the interpretation of



is turned into $([z_1, x'], \langle x'', z_2\rangle)$. Now, for each element $x \in M \oplus M$, there exist unique $z_1, x', x'', z_2 \in M$ such that $x = ([z_1, x'], \langle x'', z_2\rangle)$ (we have used the same notations for the variables in the expression and the elements of M to avoid writing the substitution explicitly). We then define ϕ so that

$$\phi(x) = \phi([z_1, x'], \langle x'', z_2\rangle) = ([0, x''], \langle x', 0\rangle).$$

The zero of $M \oplus M$ is $(0, 0) = ([0, 0], \langle 0, 0\rangle)$, hence $\phi(0, 0) = (0, 0)$, and if we take $x, y \in M \oplus M$, we decompose them as $x = ([x_1, x_2], \langle x_3, x_4\rangle)$ and $y = ([y_1, y_2], \langle y_3, y_4\rangle)$, and we have $x + y = ([x_1 + y_1, x_2 + y_2], \langle x_3 + y_3, x_4 + y_4\rangle)$, from which we obtain

$$\begin{aligned}\phi(x + y) &= ([0, x_3 + y_3], \langle x_2 + y_2, 0\rangle) = \\ &= ([0, x_3], \langle x_2, 0\rangle) + ([0, y_3], \langle y_2, 0\rangle) = \phi(x) + \phi(y),\end{aligned}$$

so ϕ is an endomorphism of $M \oplus M$. We also have

$$\mathrm{fix}(\phi) = \{([0, x], \langle x, 0\rangle) \in M \oplus M\ ;\ x \in M\},$$

which is exactly the interpretation of the above net.

All that is left to do is verifying that $\phi = \nu^\bullet$. This is proved by induction on the number m of binary cells in ν. If $m = 0$, ν is an ε-wiring, and its interpretation can be assumed without loss of generality to be generated by a balanced expression of the form

$$(\mathsf{0}, \ldots, \mathsf{0}, x_1, x_1, \ldots, x_k, x_k),$$

where the symbol $\mathsf{0}$ appears n' times, with $n' + 2k = n$ (the number of free ports of ν). Then, ϕ is the endomorphism such that

$$\phi(x_1, \ldots, x_{n'}, y'_1, y''_1, \ldots, y'_k, y''_k) = (0, \ldots, 0, y''_1, y'_1, \ldots, y''_k, y'_k),$$

which is exactly the endomorphism we introduced at p. 139 under the name $\sigma_{n',k}$, and which is equal to $\nu^\bullet$. If $m > 0$, calculations virtually identical to those of the proof of Theorem 3.44 show that we have $\phi = \nu^\bullet$ in this case as well; the details are left to the reader.

Part II

Multiport Interaction Nets and Concurrency

Chapter 4

Multiport Interaction Nets

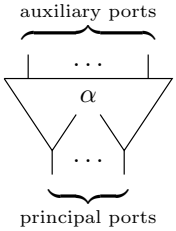
In the first part of our work we have seen how Lafont's interaction nets, and in particular the (symmetric) interaction combinators, are a very rich and interesting model of *distributed* computation. "Distributed" means that the computation evolves at several different places in a net, in a completely parallel fashion. This parallelism is in some sense "cooperative": due to the strong determinism of interaction nets, conflicts never arise during reduction, and the computation is unique up to trivial permutations of reduction rules. Therefore, no truly *concurrent* behavior can be expressed within interaction nets, where by "concurrent" we mean a situation in which computational agents do not always cooperate, but may be in competition, typically to gain access to a resource.

Multiport interaction nets are a simple concurrent extension of interaction nets, obtained by allowing cells to have more than one principal port. Multiport interaction nets have already been considered under the name of *Interaction Nets with Multiple Principal Ports* by Vladimir Alexiev in his Ph.D. thesis [Ale99], as one of several possible non-deterministic extensions of interaction nets. These systems have also been the object of Lionel Khalil's Ph.D. thesis [Kha03], in which he proved that it is actually sufficient to add to interaction nets a single cell with two principal ports and two auxiliary ports, called amb, to obtain the full power of multiport interaction nets. In spite of Khalil's result, it is still useful from the point of view of conciseness to consider cells with an arbitrary number of principal ports, as we shall do in our work.

4.1 Cells, nets

4.1.1 Multicells

In the case of multiport interaction nets, a cell is a symbol plus an arity and a *coarity*, which is the number of its principal ports:



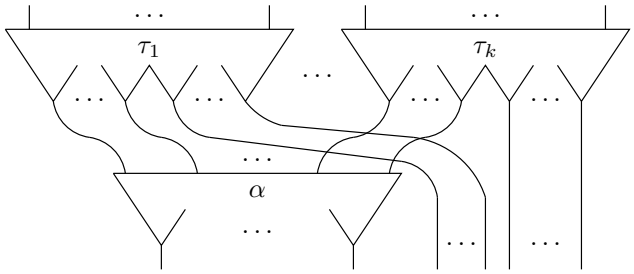
The principal ports of a cell of coarity m are supposed to be numbered from 1 to m; in the above representation, the numbering is assumed to increase from left to right. Cells whose co-arity is 1 will be called *unicells*; cells with greater co-arities will instead be called *multicells*. As usual, an alphabet is a set of cells.

Nets are naturally extended to contain multicells. The set of all nets on an alphabet Σ is still denoted by $\langle\Sigma\rangle$. The same conventions of Sect. 1.1.2 are applied to nets containing multicells.

4.1.2 Multitrees

A *multitree* is the analogue of a tree in presence of multicells. A multitree has a number of leaves (maybe zero), which is its *arity*, and also a number of roots, which can be greater than or equal to 1 and which is called its *coarity*:

- a single zeroary cell is a multitree with zero leaves and one root;
- a single wire is a multitree with one leaf and one root;
- if α is a cell of arity n and coarity m, and if $\tau_1, \ldots, \tau_k$ are multitrees of resp. arities $n_1, \ldots, n_k$ and coarities $m_1, \ldots, m_k$, such that $m_1 + \cdots + m_k \geq n$, then the net

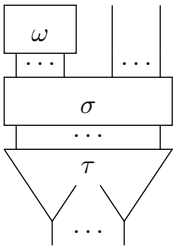


  is a multitree with $n_1 + \cdots + n_k$ leaves and m roots.

Notice that, in particular, a tree is a multitree. Multitrees will be represented just like multicells, as in the above picture.

4.1.3 Principal nets

The notion of *principal net* is adapted to the multiport framework in the most natural way: a principal net is a multitree with some leaves connected together by means of wires. In other words, a principal net of arity n and coarity m has always the following shape:

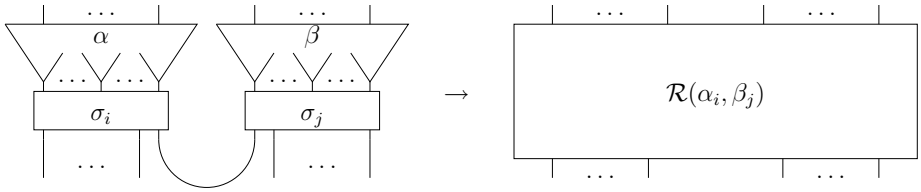


where τ is a multitree of arity $n+2k$ and coarity m, σ a permutation on $n+2k$ elements, and ω a wiring containing k wires. As usual, principal nets can be seen as "macro-cells", and will be drawn just like ordinary cells. We shall denote by $\wp(\Sigma)$ the set of principal nets built upon an alphabet Σ. Principal nets with one free port are called *packages*.

4.2 Interaction rules

4.2.1 Reduction

Just as in the single-port case, active pairs are pairs of *distinct* cells connected through one of their principal ports. Interaction rules will then be of the form



where, for graphical convenience, we have used two permutations σ_i and σ_j that "isolate" resp. the i-th principal port of α and the j-th principal port of β, i.e.

Active pairs are denoted by $\alpha^i \bowtie \beta^j$, i.e., not only the cell, but also the principal ports are taken into account. As usual, the right member of an interaction rule must be a net respecting the interface of the active pair, i.e., there must be a bijection between the free ports of the right and left members of each rule; this bijection will always be clear from the graphical representation.

Much like in the single-port case, the active pair under reduction can be "disconnected" from the net and be replaced by its reduct. This is the basic rewriting step, and it is written $\mu \to \mu'$ (μ reduces in one step to μ'). We denote by $\to^*$ the reflexive and transitive closure of $\to$, and if $\mu \to^* \mu'$, we say that μ' is a *reduct* of μ.

Notice that a multicell can in general be involved in several active pairs; the choice of which one is reduced is non-deterministic. An immediate consequence of this fact is that we lose the confluence property which characterizes interaction nets; but of course this is precisely what we are aiming at.

There are some additional constraints on interaction rules: first of all, we keep the requirement that there may be *at most* one rule for each pair of principal ports; allowing several rules, to be chosen non-deterministically, only complicates the definition without adding expressive power. We also observe that, just as in the single-port case, active pairs intrinsically lack an orientation, so the reduct $\mathcal{R}(\beta_j, \alpha_i)$ must be essentially the same as $\mathcal{R}(\alpha_i, \beta_j)$, just "flipped over", i.e., $\mathcal{R}(\beta_j, \alpha_i) = \overline{\mathcal{R}(\alpha_i, \beta_j)}$, according to the notation introduced in Sect. 1.1.2. Moreover, reducts cannot contain active pairs.

4.2.2 Multiport interaction net systems

We are now ready to introduce the formal definition of a multiport Interaction Net System (mINS):

Definition 4.1 (Multiport Interaction Net System (mINS)) *A* multiport Interaction Net System $\mathcal{S}$ *is a couple* $(\Sigma, \mathcal{R})$*, where:*

- Σ *is an alphabet, possibly denumerably infinite;*
- $\mathcal{R}$ *is a partial function taking an active pair and yielding a net without active pairs having the same interface, such that, if* $\mathcal{R}(\alpha_i, \beta_j)$ *is defined, then* $\mathcal{R}(\beta_j, \alpha_i) = \overline{\mathcal{R}(\alpha_i, \beta_j)}$*;*

A mINS is said to be finite *or* infinite *according to the cardinality of its alphabet.*

mINS's will always be assumed to be finite; however, in Chapter 5 we shall also use infinite systems (this is why they have been allowed in the definition), in which case we shall always specify it. If the maximum coarity of the cells of a system $\mathcal{S}$ is n, we say that $\mathcal{S}$ is a nINS; if $n = 1$, we omit it and we just say that it is an INS, i.e., a "traditional" interaction net system.

4.2.3 Types

As done in Sect. 1.2.3, we can endow mINS's with a *type discipline*: given a system $\mathcal{S}$, we consider a set of *constant types*, ranged over by T, and to each port of the cells of $\mathcal{S}$ we assign an *input type* (T^-) or an *output type* (T^+). We say that a net is *well typed* if inputs are connected to outputs of the same type; a rule is well typed if both its left and right members are well typed, and the typing of the interface is preserved. If all rules of $\mathcal{S}$ are well typed, and if every well typed cut has a corresponding rule, we say that $\mathcal{S}$ is a *typed* mINS.

In a typed mINS, it is sometimes useful to have *overloaded* cells, i.e., cells which admit more than one typing for their ports; the typical example is a *duplicator cell*, which can duplicate no matter what and must therefore be capable of interacting with any cell of any type (see Fig. 5.7 and 5.8 in Sect. 5.3).

The type discipline will be used to guarantee certain correctness properties of the system, mainly that "unreasonable" active pairs never arise through reduction, like, say, that a cell representing integer addition never interacts with a string constructor.

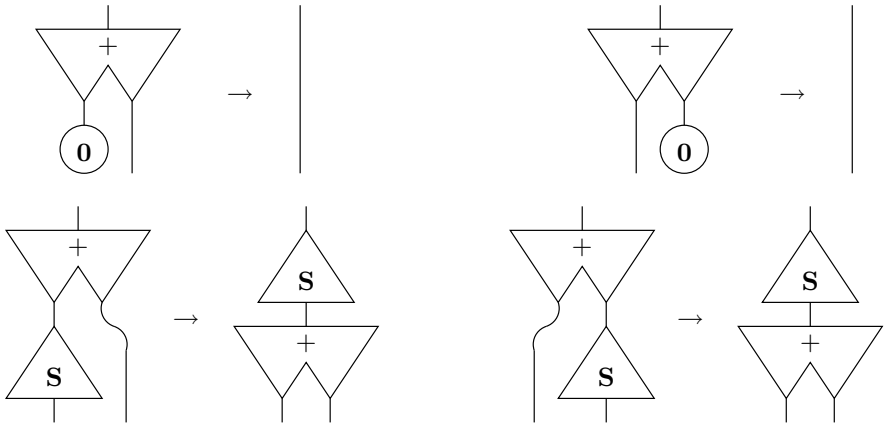
4.3 Examples

As an example of the expressive power gained by allowing more than one principal port, we show how a few simple parallel algorithms can be encoded using multiport interaction nets.

First of all, consider the *parallel or* cell, with the following reduction rules:

The cells ff and tt represent the Boolean truth values (resp. "false" and "true"), while the cell ε is an eraser. Clearly, the por cell computes the disjunction of two truth values in a parallel fashion: in case one of the two arguments is tt, it returns tt without looking at the other.

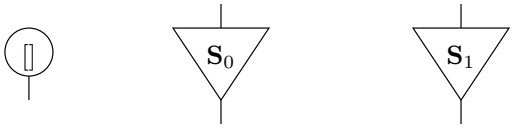
The same "parallelization" can be applied to the + cell of the INS introduced in Sect. 1.3.3:



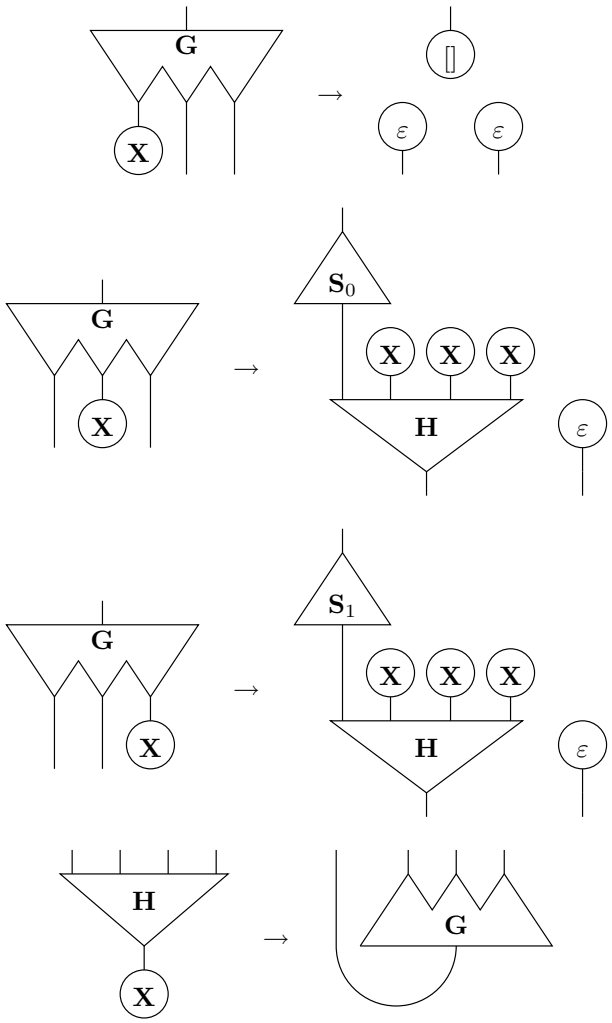
It is clear that what this multiport version of the + cell really does is merging two streams: in fact, we have just implemented the so-called *bottom-avoiding merge* (see for example Andrew Moran's Ph.D. thesis [Mor98] for an account of the different forms of non-deterministic merging of infinite streams).

Bottom-avoiding merge is an algorithm taking two (possibly infinite) streams of data x and y and returning a stream z which results from randomly merging x and y, fullfilling the following condition: if one of x or y is finite, then every element of the other stream appears in z. This stream-merging algorithm is called "bottom-avoiding" because it is able to avoid non-termination caused by partially defined streams. It is different from *fair merge* though, because, in case both x and y are infinite, it may happen that the output contains only elements of one of the two streams. In fact, it is impossible to implement fair merge in multiport interaction nets: this follows from Khalil's proof that multiport interaction nets are equivalent to interaction nets plus McCarthy's amb [Kha03], composed with a well-known result of Panangaden and Shanbogue [PS88].

Since we are speaking of streams, we can show how a random stream generator can be implemented. Suppose we have the cells

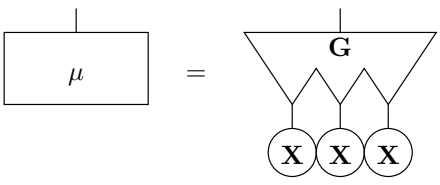


If we take [] to mean "empty list", and $\mathbf{S}_0, \mathbf{S}_1$ to mean resp. "0" and "1", we can define binary lists much in the same way we defined unary integers in Sect. 1.3.3. Now consider the cells **X**, **G**, and **H**, with the following interaction rules:

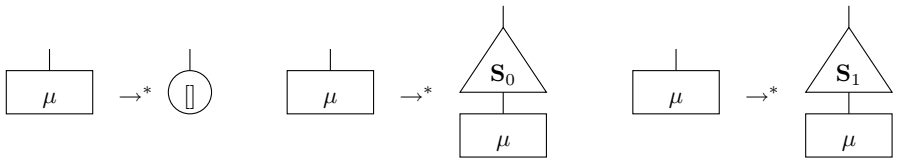


where ε is the usual eraser cell.

Now if we put



we invite the reader to check that the net μ admits the following three reductions:



Therefore, μ is able to non-deterministically generate any (possibly infinite) 2-bit stream.

Chapter 5

Encoding the π-calculus

In this chapter we show that multiport interaction nets are an expressive model of concurrent computation by encoding the full π-calculus in them.

A considerable number of graphical representations of the π-calculus (or other process calculi) can be found in the existing literature. Let us mention for example Robin Milner's π-nets [Mil94], Joachim Parrow's Interaction Diagrams [Par95], and Yuxi Fu's Reaction Graphs [Fu98]. All these approaches succeed in describing concurrent dynamics as graph rewriting, but the treatment of prefixing is not very natural (in π-nets and Reaction Graphs, some form of "guarded box" is used, while Interaction Diagrams use polyadicity to encode causal dependency), and they all need boxes to represent replication, so that duplication is seen as a synchronous, global operation. It must also be observed that none of the existing graphical representations is ever shown to cover the π-calculus in all of its features, including sums and match prefix.

More recently, Cosimo Laneve, Parrow, and Björn Victor proposed Solo Diagrams [LPV01] as a graphical presentation of the *solos calculus* [LV99]. They too use replication boxes, but show that these can be limited to certain configurations which ensure constant-time reductions, and thus locality.

Much closer to the spirit of interaction nets, a nice graphical representation of (an extension of) the *fusion calculus* has been given by Emmanuel Beffara and François Maurel [BM05], in which nevertheless replication must be accommodated using boxes. In view of our results, it does not seem unlikely that multiport interaction nets can provide both an alternative, "box-less" graphical encoding for the solos calculus and a purely local version of Beffara and Maurel's Concurrent Nets.

It is worth mentioning the comparison between Interaction Nets and concurrent systems done by Nobuko Yoshida [Yos95], who found that, when seen from the point of view of Interaction Nets, the graphical representation for her concurrent combinators amounts more or less to allow *hyperwires* connecting cells, i.e., wires that link together more than two ports. This is also explicitly seen in Beffara and Maurel's Concurrent Nets. As a matter of fact, our approach "internalizes" these hyperconnections, extending Lafont's systems not with respect to the topology of the connections between cells but to the nature of the cells themselves.

As already recalled, multiport interaction nets have been considered by Vladimir Alexiev in his Ph.D. thesis [Ale99]. One of his main results is that this

extension of interaction nets is as expressive as the "hyperwire" extension mentioned above. Alexiev also gave a graphical encoding of the finite π-calculus in (a variant of) multiport interaction nets, leaving open the problem of extending it to replication.

Our encoding (quite different from Alexiev's one, even in the finite case) covers every single feature of the π-calculus, in particular replication, which is crucial in terms of expressiveness. Compared to the aforementioned graphical formalisms, ours has an exceptional advantage: no "box" or other global notion is needed, i.e., the dynamics is fully local. In other words, our encoding may be seen as the equivalent of *sharing graphs* for the λ-calculus. In perspective, this opens the possibility for a new semantical study of concurrency, as the algebraic semantics enjoyed by Lafont's original systems (the geometry of interaction, cf. Sect. 3.3) might be extended to multiport interaction nets (this is not developed here though).

We also stress the fact that, unlike virtually any other graphical system proposed for concurrency, multiport interaction nets *are not* built around the π-calculus, or any other process calculus. On the contrary, they ought to be seen as an independent, alternative model of concurrency, which is shown here to be equivalent to the π-calculus; the results of this chapter should be read more in this sense than as "yet-another-graphical-representation-of-π".

5.1 The π-calculus

In this section we briefly recall the main definitions concerning the π-calculus. Our main reference will be Sangiorgi and Walker's book [SW01], even though we choose a slightly different presentation.

5.1.1 Processes

Given a denumerable set of *names*, ranged over by $x, y, z, \ldots$, the *prefixes*, *extended prefixes*, and *processes* of the π-calculus are resp. generated by the following grammars:

$$\begin{array}{rcl} \pi & ::= & \overline{x}y \mid x(z) \mid \tau \mid [x = y] \\ \kappa & ::= & \overline{x}(z) \mid \pi \\ P, Q & ::= & \displaystyle\sum_{i \in I} \pi_i.P_i \mid P \mid Q \mid \boldsymbol{\nu}(z)P \mid \kappa \rhd P \end{array}$$

Processes of the form $\sum_{i \in I} \pi_i.P_i$, where I is a finite set of indexes, are called *summations*; if $I = \emptyset$, the summation is denoted $\mathbf{0}$. Summations are ranged over by $M, N, \ldots$; if $M = \sum_{i \in I} \pi_i.P_i$ and there exists an $i \in I$ such that $\pi.P = \pi_i.P_i$, we say that $\pi.P$ is a *summand* of M, and write $\pi.P \in M$.

Names must be seen as communication channels and, at the same time, as the information which is transmitted on such channels. The (extended) prefixes $x(z)$, $\overline{x}(z)$, and the operator $\boldsymbol{\nu}(z)$ are all binding constructors for the name z; we can define the set $\mathsf{fn}(P)$ of the *free names* of a process P as follows:

- $\mathsf{fn}(\sum_{i \in I} \pi_i.P_i) = \bigcup_{i \in I} \mathsf{fn}(P_i) \setminus B_i$, where $B_i = \{z\}$ if $\pi_i = x(z)$, or $B_i = \emptyset$ otherwise;

- $\mathsf{fn}(P \mid Q) = \mathsf{fn}(P) \cup \mathsf{fn}(Q)$;
- $\mathsf{fn}(\boldsymbol{\nu}(z)P) = \mathsf{fn}(P) \setminus \{z\}$;
- $\mathsf{fn}(\kappa \rhd P) = \mathsf{fn}(P) \setminus B$, where $B = \{z\}$ if $\kappa = x(z)$ or $\kappa = \overline{x}(z)$, or $B = \emptyset$ otherwise.

We shall always consider processes modulo α-equivalence, i.e., modulo capture-free renaming of bound names. Whenever $z \in \mathsf{fn}(P)$, we denote by $P\{y/z\}$ the process in which all occurrences of z in P have been replaced by y.

We now give a quick informal description of the meaning of the various constructions used for building processes:

- Summations express the simultaneous availability of a finite number of capabilities, at most one of which can be exerted. Upon exertion of a capability, the other capabilities of the summation are lost. These capabilities are expressed by the prefixes:
  - $\overline{x}y$ is the *output prefix*: a summation containing a summand of the form $\overline{x}y.P$ has the capability of sending the name y over the name x, and then proceed as P;
  - $x(z)$ is the *input prefix*: a summation containing a summand of the form $x(z).P$ has the capability of receiving a name y over the name x, and then proceed as $P\{y/z\}$;
  - τ is the *internal action prefix*: a summation containing a summand of the form $\tau.P$ has the capability of silently evolving to P;
  - $[x = y]$ is the *match prefix*: a summation containing a summand of the form $[x = y].P$ has the capability of evolving to P in case $x = y$.

  The empty summation $\mathbf{0}$ represent the inert process, i.e., a process that can do nothing. Notice that, if we ignore the presence of a dummy free name z, the internal action prefix can be seen as a shorthand notation for the prefix $[z = z]$.
- The *parallel composition* $P \mid Q$ denotes a process in which the two subprocesses P and Q are free to evolve in parallel, possibly interacting with each other.
- The *restriction operator* $\boldsymbol{\nu}(z)$ makes the name z private to a subprocess, i.e., in $\boldsymbol{\nu}(z)P$, z is known only within P.
- The *replicated prefix* construction adds the possibility of infinite behavior, depending on the extended prefix used:
  - $\overline{x}(z)$ is called the *bound output prefix*: the process $\overline{x}(z) \rhd P$ acts like a "server" capable of indefinitely sending fresh names over the channel x (i.e., names that will be private to the receiver) and then proceed each time as a new instance of P;
  - the process $\pi \rhd P$ can indefinitely exert the capability expressed by π, and then proceed each time as a new instance of P (modulo a substitution in case π is an input prefix).

$$
\begin{array}{rcl}
P \mid (Q \mid R) & \equiv & (P \mid Q) \mid R \\
P \mid Q & \equiv & Q \mid P \\
P \mid \mathbf{0} & \equiv & P \\
\\
\boldsymbol{\nu}(z)\boldsymbol{\nu}(w)P & \equiv & \boldsymbol{\nu}(w)\boldsymbol{\nu}(z)P \\
\boldsymbol{\nu}(z)\mathbf{0} & \equiv & \mathbf{0} \\
\boldsymbol{\nu}(z)(P \mid Q) & \equiv & P \mid \boldsymbol{\nu}(z)Q, \text{ if } z \notin \mathsf{fn}(P)
\end{array}
$$

Figure 5.1: The axioms defining structural congruence.

The above informal description will be formalized in the forthcoming sections. Before that, we stress for the reader already acquainted with the π-calculus the key differences between our presentation and more standard ones (like that of Sangiorgi and Walker [SW01]):

- Instead of using (the more general) plain replication (i.e., processes of the form $!P$), we have chosen replicated prefixes as done for example by Pierce and Turner for `Pict` [PT97]. It is well known that the expressive power of the calculus is not limited by such choice. In the more standard sintax using full replication, our processes $\overline{x}y \rhd P$, $x(z) \rhd P$, and $\overline{x}(z) \rhd P$ would be written resp. as $!\overline{x}y.P$, $!x(z).P$, and $!\boldsymbol{\nu}(z)\overline{x}z.P$.
- Our match prefixes are "real" prefixes, as opposed to the tradition in which matching is usually added by including the production $\pi ::= [x = y]\pi$ in the grammar generating prefixes, with the intended meaning that in $[x = y]\pi.P$, π is not activated until x and y match. This is normally rendered by adding to the definition of structural congruence the axiom $[x = x]\pi.P \equiv \pi.P$. In this syntax, our presentation is equivalent to allowing matchings only before internal action prefixes, like $[x = y]\tau.P$; by the way, $[x = y]\tau.\pi.P$ and $[x = y]\pi.P$ are full bisimilar (though not strongly), so our choice does not alter the expressiveness of the calculus in a sensitive way.

Both of the above differences are technically motivated by the good properties we want our encoding to enjoy with respect to structural congruence: without them, Proposition 5.3 would not hold.

5.1.2 Reduction

A first understanding of the meaning of π-calculus processes comes by looking at their dynamics. The π-calculus can in fact be seen as a rewriting system, in which processes evolve by exchanging names along... names, i.e., channels.

We start by introducing *structural congruence*, denoted by $\equiv$, which is the reflexive, symmetric, transitive, and contextual closure of the relation generated by the axioms of Fig. 5.1. Structurally congruent processes must be considered completely equivalent, i.e., if $P \equiv Q$, then P and Q are morally two notations for the same process. The first three axioms state that the set of processes is a

monoid with respect to parallel composition, with neutral element $\mathbf{0}$; thanks to this, we shall always omit useless parentheses when writing parallel compositions of three or more processes. The other three axioms concern name restriction, and state resp. that

- the order in which one restricts names is irrelevant; because of this, we shall use the notation $\boldsymbol{\nu}(z_1, \ldots, z_n)P$ for $\boldsymbol{\nu}(z_1) \ldots \boldsymbol{\nu}(z_n)P$, or more compactly $\boldsymbol{\nu}(\widetilde{z})P$, where $\widetilde{z}$ denotes a set of names;
- restricting a name on the empty summation has no effect;
- restriction commutes to parallel composition, as long as we use α-equivalence to avoid name-capturing. This axiom, called *scope extrusion*, is of fundamental importance in the π-calculus, since it allows the scope of a restriction to be enlarged through communication, as we shall see in a moment.

As an example, consider a process Q such that $z \notin \mathsf{fn}(Q)$, and consider the process $P = \boldsymbol{\nu}(z)Q$. By neutrality of $\mathbf{0}$, we have $P \equiv P \mid \mathbf{0}$; by scope extrusion, we have $P \mid \mathbf{0} \equiv Q \mid \boldsymbol{\nu}(z)\mathbf{0}$, and by the second axiom concerning restriction we obtain that $\boldsymbol{\nu}(z)Q \equiv Q$, i.e., restricting a name on a process not containing it (or in which the name is already bound) has no effect.

We invite the reader to check the following result, which gives a sort of "canonical form" for π-calculus processes:

Proposition 5.1 *Any process is structurally congruent to a process of the form*

$$\boldsymbol{\nu}(\widetilde{z})(M_1 \mid \ldots \mid M_m \mid R_1 \mid \ldots \mid R_n),$$

where the M_i are summations, and the R_i are of the form $\kappa \rhd Q_i$.

The reduction relation $\longrightarrow$ is defined by the inference rules of Fig. 5.2. These rules agree with the informal description given above: processes exchange names by synchronizing on a channel, and replicated processes "stay there" after interacting. The last three rules simply say that reduction can be performed in the context of a parallel composition or of a restriction, and that the reduction relation is defined modulo structural equivalence.

Let us give a few examples illustrating the reduction relation just introduced. Set $P = \overline{x}y.\mathbf{0}$ and $Q = x(z).(z(w).Q_1 \mid \overline{z}a.\mathbf{0})$, where $y \notin \mathsf{fn}(Q_1)$, and consider the process $R = P \mid Q \mid \overline{y}b.\mathbf{0}$. The rules of Fig. 5.2 allow us to derive that

$$R \longrightarrow y(w).Q_1 \mid \overline{y}a.\mathbf{0} \mid \overline{y}b.\mathbf{0} = R_1.$$

At this point, both the reduction

$$R_1 \longrightarrow Q_1\{a/w\} \mid \overline{y}b.\mathbf{0}$$

and the reduction

$$R_1 \longrightarrow Q_1\{b/w\} \mid \overline{y}a.\mathbf{0}$$

are possible, which shows that reduction is not confluent in general, as Q_1 may be for example structurally congruent to $\mathbf{0}$, while the processes $\overline{y}a.\mathbf{0}$ and $\overline{y}b.\mathbf{0}$ are not structurally congruent and do not admit further reductions.

$$\frac{\overline{x}y.P \in M \qquad x(z) \in N}{M \mid N \longrightarrow P \mid Q\{y/z\}}$$

$$\frac{\tau.P \in M}{M \longrightarrow P} \qquad \frac{[x = x].P \in M}{M \longrightarrow P}$$

$$\frac{\overline{x}y.P \in M \qquad R = x(z) \rhd Q}{M \mid R \longrightarrow P \mid Q\{y/z\} \mid R} \qquad \frac{R = \overline{x}y \rhd P \qquad x(z).Q \in M}{R \mid M \mid \ \longrightarrow R \mid P \mid Q\{y/z\}}$$

$$\frac{R = \overline{x}(z) \rhd P \qquad x(z).Q \in M}{R \mid M \longrightarrow R \mid \boldsymbol{\nu}(z)(P \mid Q)}$$

$$\frac{R = \overline{x}y \rhd P \qquad S = x(z) \rhd Q}{R \mid S \longrightarrow P \mid Q\{y/z\} \mid R \mid S} \qquad \frac{R = \overline{x}(z) \rhd P \qquad S = x(z) \rhd Q}{R \mid S \longrightarrow \boldsymbol{\nu}(z)(P \mid Q) \mid R \mid S}$$

$$\frac{P \longrightarrow P'}{P \mid Q \longrightarrow P' \mid Q} \qquad \frac{P \longrightarrow P'}{\boldsymbol{\nu}(z)P \longrightarrow \boldsymbol{\nu}(z)P'}$$

$$\frac{P \equiv Q \qquad Q \longrightarrow Q' \qquad Q' \equiv P'}{P \longrightarrow P'}$$

Figure 5.2: Rules defining reduction in the π-calculus.

Suppose now that instead of using P we use the process $P' = \boldsymbol{\nu}(y)P$, forming $R' = P' \mid Q \mid \overline{y}b.\mathbf{0}$. Since $y \notin \mathsf{fn}(Q)$ we can write $R' \equiv \boldsymbol{\nu}(y)(P \mid Q) \mid \overline{y}b.\mathbf{0}$, or, using α-convertion and more scope extrusion, $R' \equiv \boldsymbol{\nu}(t)(P\{t/y\} \mid Q \mid \overline{y}b.\mathbf{0})$. Now P and Q can interact as before, but this time we obtain

$$R' \longrightarrow \boldsymbol{\nu}(t)(t(w).Q_1 \mid \overline{t}a.\mathbf{0} \mid \overline{y}b.\mathbf{0}) = R'_1.$$

Here the situation has completely changed: the process $\overline{y}b.\mathbf{0}$ can no longer interfere with the two subprocesses of Q, which are now free to communicate on the private channel t. The only reduction possible is therefore

$$R'_1 \longrightarrow Q_1\{a/w\} \mid \overline{y}b.\mathbf{0},$$

where the restriction on t has disappeared because the above process does not contain any free occurrence of it.

This is an example of the power of scope extrusion: at first, the restriction on y is effective only on P, but because this name is sent to Q, the scope of the restriction is enlarged to cover the subprocesses of Q. More generally, before two processes of the form $\boldsymbol{\nu}(y)\overline{x}y.P$ and $x(z).Q$ interact, the name y is known only within $\overline{x}y.P$; by α-conversion, we can always suppose that $y \notin \mathsf{fn}(Q)$, so after the interaction, the result is $\boldsymbol{\nu}(y)(P \mid Q\{y/z\})$, in which y is now known to Q as well, and can be used to communicate privately with P.

Set now $P = \overline{x}a.\mathbf{0} + \overline{x}b.\mathbf{0}$, and $Q = x(z).Q_1$. The process P has simultaneously the capability of sending a or b over the channel x, but only one of these two capabilities can be exerted, at the expense of the other. As a consequence, the process $P \mid Q$ has two reductions, depending on which component of the summation is chosen to synchronize with Q:

$$P \mid Q \longrightarrow Q_1\{a/z\},$$

and

$$P \mid Q \longrightarrow Q_1\{b/z\}.$$

In both cases, the component of the summation not used in the interaction simply disappears.

One last example to illustrate the use of replicated prefixes. Put $R = \overline{x}y \triangleright \mathbf{0}$, and consider the process

$$P = x(z).Q_1 \mid x(w).Q_2 \mid R.$$

P represents a situation in which two processes are listening on channel x for a name to be sent. If R were of the form $\overline{x}y.\mathbf{0}$, only the request of one of them could be satisfied; fortunately for them, R has a replicated prefix, which means that it can send as many names as there are processes requiring them:

$$P \longrightarrow Q_1\{y/z\} \mid x(w).Q_2 \mid R \longrightarrow Q_1\{y/z\} \mid Q_2\{y/w\} \mid R.$$

Notice that y is shared by Q_1 and Q_2, and therefore these two processes may communicate with each other through it. This could be unwanted, i.e., there are cases in which a communication between Q_1 and Q_2 would be regarded as an interference. The bound output prefix is used to make sure that no interference is produced, i.e., that the names sent remain private within Q_1 and Q_2. In fact, if we set $R' = \overline{x}(y) \triangleright \mathbf{0}$, we have

$$\begin{aligned} x(z).Q_1 \mid x(w).Q_2 \mid R' &\longrightarrow \boldsymbol{\nu}(y_1)(Q_1\{y_1/z\} \mid x(w).Q_2 \mid R') \longrightarrow \\ &\longrightarrow \boldsymbol{\nu}(y_1, y_2)(Q_1\{y_1/z\} \mid Q_2\{y_2/w\} \mid R'). \end{aligned}$$

$$\frac{\overline{x}y.P \in M}{M \xrightarrow{\overline{x}y} P} \qquad \frac{x(z).P \in M}{M \xrightarrow{xy} P\{y/z\}} \qquad \frac{P \xrightarrow{\overline{x}z} P'}{\boldsymbol{\nu}(z)P \xrightarrow{\overline{x}(z)} P'} \quad z \neq x$$

$$\frac{R = \overline{x}y \rhd P}{R \xrightarrow{\overline{x}y} P \mid R} \qquad \frac{R = x(z) \rhd P}{R \xrightarrow{xy} P\{y/z\} \mid R} \qquad \frac{R = \overline{x}(z) \rhd P}{R \xrightarrow{\overline{x}(z)} P \mid R} \quad z \neq x$$

$$\frac{\tau.P \in M}{M \xrightarrow{\tau} P} \qquad \frac{[x = x].P \in M}{M \xrightarrow{\tau} P}$$

$$\frac{P \xrightarrow{\overline{x}y} P' \qquad Q \xrightarrow{xy} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} * \qquad \frac{P \xrightarrow{\overline{x}(z)} P' \qquad Q \xrightarrow{xz} Q'}{P \mid Q \xrightarrow{\tau} \boldsymbol{\nu}(z)(P' \mid Q')} * \quad z \notin \mathsf{fn}(Q)$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} * \quad \mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \emptyset \qquad \frac{P \xrightarrow{\alpha} P'}{\boldsymbol{\nu}(z)P \xrightarrow{\alpha} \boldsymbol{\nu}(z)P'} \quad z \notin \mathsf{n}(\alpha)$$

Figure 5.3: The inference rules defining the transitions of the π-calculus. Three rules are omitted, corresponding to the symmetric versions of the rules marked with a $*$, in which the order of P and Q is exchanged in the parallel composition.

5.1.3 Transitions and behavioral equivalence

In contrast to deterministic systems like the λ-calculus, what is more interesting about the π-calculus is not the reduction relation defined above, but rather the notion of *transition*. The π-calculus can in fact be seen as a labelled transition system, according to the following definition:

Definition 5.1 (Actions, transitions) *The* actions *of the π-calculus are generated by the following grammar:*

$$\alpha ::= \overline{x}y \mid xy \mid \overline{x}(z) \mid \tau.$$

The four actions are called respectively output action*,* input action*,* bound output action*, and* invisible action*.*

The set of names appearing in an action α is denoted by $\mathsf{n}(\alpha)$; the set of bound names *of an action α, denoted by $\mathsf{bn}(\alpha)$, is defined to be equal to $\{z\}$ in case $\alpha = \overline{x}(z)$, and empty otherwise. The name x in the actions $\overline{x}y$, xy, and $\overline{x}(y)$ is called the* subject *of the action.*

The labelled transitions *of the π-calculus are generated by the inference rules given in Fig. 5.3.*

Seeing the π-calculus as a labelled transition system is much more convenient than seeing it as a rewriting system. The following result tells us that nothing is lost in this change of viewpoint, since reduction is exactly captured by invisible transitions modulo structural congruence:

Lemma 5.2 (Harmony Lemma) *$P \longrightarrow P'$ iff $P \xrightarrow{\tau} \equiv P'$.*

PROOF. See Sangiorgi and Walker's Lemma 1.4.15, p. 51 [SW01]. □

In particular, the Harmony Lemma justifies why reduction is rarely considered when dealing with the π-calculus. As a matter of fact, only little mention of it will be made in the rest of our work.

In our encoding of the π-calculus into multiport interaction nets, a special rôle will be played by what we call fully invisible transitions:

Definition 5.2 (Fully invisible actions) *We say that a process P is capable of evolving to Q through a* fully invisible action*, $P \xrightarrow{\tilde{\tau}} Q$, if $P \xrightarrow{\tau} Q$ and the subject name used in the transition (if there is one) is under the scope of a restriction.*

For example, any process of the form $\tau.P$ has a fully invisible transition to P. Instead, if we put $P = \overline{z}a.Q_1 \mid z(w).Q_2$ and $P' = Q_1 \mid Q_2\{a/w\}$, while having $P \xrightarrow{\tau} P'$, we do not have $P \xrightarrow{\tilde{\tau}} P'$; on the contrary, $\boldsymbol{\nu}(z)P \xrightarrow{\tilde{\tau}} \boldsymbol{\nu}(z)P'$.

Actions and transitions are fundamental to define the behavioral equivalences which are of most interest for the study of the π-calculus. In particular, we recall here the definition of *strong barbed congruence*, which, even though considered to be too discriminative to be a satisfying behavioral equivalence, is the basis for the definition of (weak) *barbed congruence*, arguably the most natural behavioral equivalence for the π-calculus.

In the following, we use the term *coname* to denote an object of the form $\overline{x}$, where x is a name; we use the symbol x to range over names and conames.

Definition 5.3 (Observability predicates) *Let P be a process. We say that a name x is* observable *in P, notation $P{\downarrow_x}$, iff $P \xrightarrow{xy} P'$ for some y and some P'. Similarly, we say that a coname $\overline{x}$ is observable in P, notation $P{\downarrow_{\overline{x}}}$, iff $P \xrightarrow{\overline{x}y} P'$ or $P \xrightarrow{\overline{x}(z)} P'$ for some y, z, P'.*

Definition 5.4 (Strong barbed bisimilarity) Strong barbed bisimilarity*, denoted $\dot{\sim}$, is the greatest symmetric relation on π-calculus process such that, whenever $P \dot{\sim} Q$,*

1. *$P{\downarrow_{\mathsf{x}}}$ implies $Q{\downarrow_{\mathsf{x}}}$;*
2. *if $P \xrightarrow{\tau} P'$, then $Q \xrightarrow{\tau} \dot{\sim} P'$.*

Strong barbed congruence is the context closure of strong barbed bisimilarity. We shall not formally define what a π-calculus context is; we leave the meaning to the intuition of the reader.

Definition 5.5 (Strong barbed congruence) *Two processes P, Q are said to be* strong barbed congruent*, notation $P \simeq^{\mathrm{c}} Q$, iff, for any context C, $C[P] \dot{\sim} C[Q]$.*

The faithfulness of our encoding of the π-calculus into multiport interaction nets will be proved by establishing a sort of strong barbed bisimilarity between processes and their translation, after having introduced the notions corresponding to observability and (fully) invisible transitions for the nets representing such translations.

5.1.4 Subcalculi

In the forthcoming sections we shall consider two subcalculi of the full π-calculus, which will be useful to "modularize" our encoding and present it step by step, in a complexity-increasing fashion.

The simplest subcalculus is what we call the *finite π-calculus*, or Fπ. It only models name-passing and name-restriction, without any other construct (notably without either replication or "true" summations). The prefixes and processes of Fπ are resp. generated by the following grammars:

$$\begin{array}{rcl} \pi & ::= & \overline{x}y \mid x(z) \\ P, Q & ::= & \mathbf{0} \mid \pi.P \mid P \mid Q \mid \boldsymbol{\nu}(z)P \ . \end{array}$$

With respect to the full calculus, apart from the exclusion of internal action and match prefixes, the biggest restriction made on Fπ is the exclusion of replicated prefixes, which implies the elimination of extended prefixes as well. Moreover, summations are restricted to the case in which the set I on which the sum ranges over is either empty or a singleton. The lack of replication makes Fπ "finite" in the sense that only finite dynamics can be modelled in it.

A more complex subcalculus, in which Fπ is strictly included, is what we call the "core" π-calculus, or Cπ, which is obtained from the full calculus by ignoring internal action and match prefixes, and by restricting again summations to range over the empty set or a singleton. The prefixes, extended prefixes, and processes of Cπ are resp. generated by

$$\begin{array}{rcl} \pi & ::= & \overline{x}y \mid x(z) \\ \kappa & ::= & \overline{x}(z) \mid \pi \\ P, Q & ::= & \mathbf{0} \mid \pi.P \mid P \mid Q \mid \boldsymbol{\nu}(z)P \mid \kappa \rhd P \end{array}$$

Cπ can be seen as an extended, synchronous version of Pierce and Turner's `Pict` [PT97]; as such, not only is it Turing-complete, but is also expressive enough to be able to model virtually any concurrent system arising in practice. Exhibiting a faithful encoding of Cπ into a concurrent formalism is thus a fairly strong proof of its expressive power.

5.2 Encoding the finite π-calculus

Our first step will be to find a mINS which implements Fπ, as introduced in Sect. 5.1.4. Consider the (infinite) typed mINS $\mathcal{F}_\infty$ whose alphabet and rules are given resp. in Fig. 5.4 and Fig. 5.5. Types have been omitted in the rules, but the reader can check that they are all well typed. The first rule of Fig. 5.5 is an example of "template rule": for a fixed $m \geq 0$, it actually condenses $2(m+1)$ rules. Template rules, the fact that ϵ ranges over $\{+, -\}$, and the permutations σ^ϵ will be notational conventions constantly adopted throughout the rest of the chapter.

We now give a brief description of the rôle played by the various components of the system:

- Types N, C, and Q represent resp. *names*, *continuations*, and *queues*.

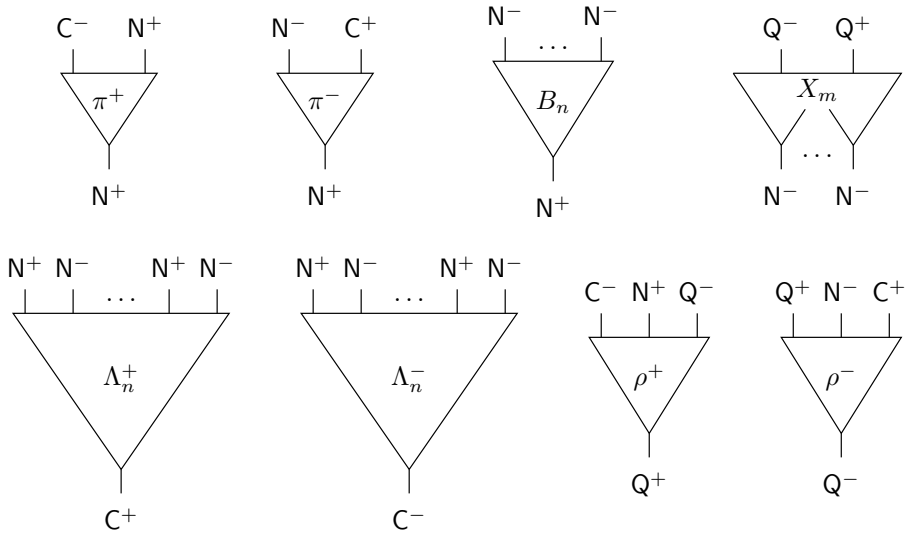


Figure 5.4: The alphabet of $\mathcal{F}_\infty$

- π^+ and π^- cells will implement resp. output and input prefixes; each of them is ready to make a request on a name, and bears a continuation and another name (either the name being sent or the place-holder for substitution).
- B_n is a monocell with n auxiliary ports, with $n \geq 0$; this family of cells will be needed to bind the occurrences of the name used by an input prefix.
- X_m is a cell with m principal ports ($m \geq 1$) and 2 auxiliary ports. We stipulate that X_0 is just a wire of type Q. The cells belonging to this family will implement names: they are capable of concurrently handling several requests coming from π^+ and π^- cells, and they have two FIFO queues (one for inputs, the other for outputs) that will be used to store prefixes waiting for interaction. They also handle requests from B_n cells; the interaction with these cells models name-passing and the associated substitution.
- Λ_n^+ and Λ_n^- are monocells of arity $2n$, $n \geq 0$. These two families will implement the blocking function of prefixes: they "suspend" a connection until they interact.
- ρ^+ and ρ^- cells will be needed to represent resp. output and input queues on channels; they bear the same information as π^ϵ cells, plus a pointer to the rest of the queue. Their interaction synchronizes two prefixes: the name being sent is connected to the place-holder for substitution, and their two continuations are connected, so that they can be unblocked.

We shall now define a translation $\langle\cdot\rangle$ of Fπ processes into $\mathcal{F}_\infty$ nets. This encoding enjoys a clear correspondence with the interactive structure of processes, but accounts only for certain kinds of transitions. The free ports of $\langle P\rangle$ will be

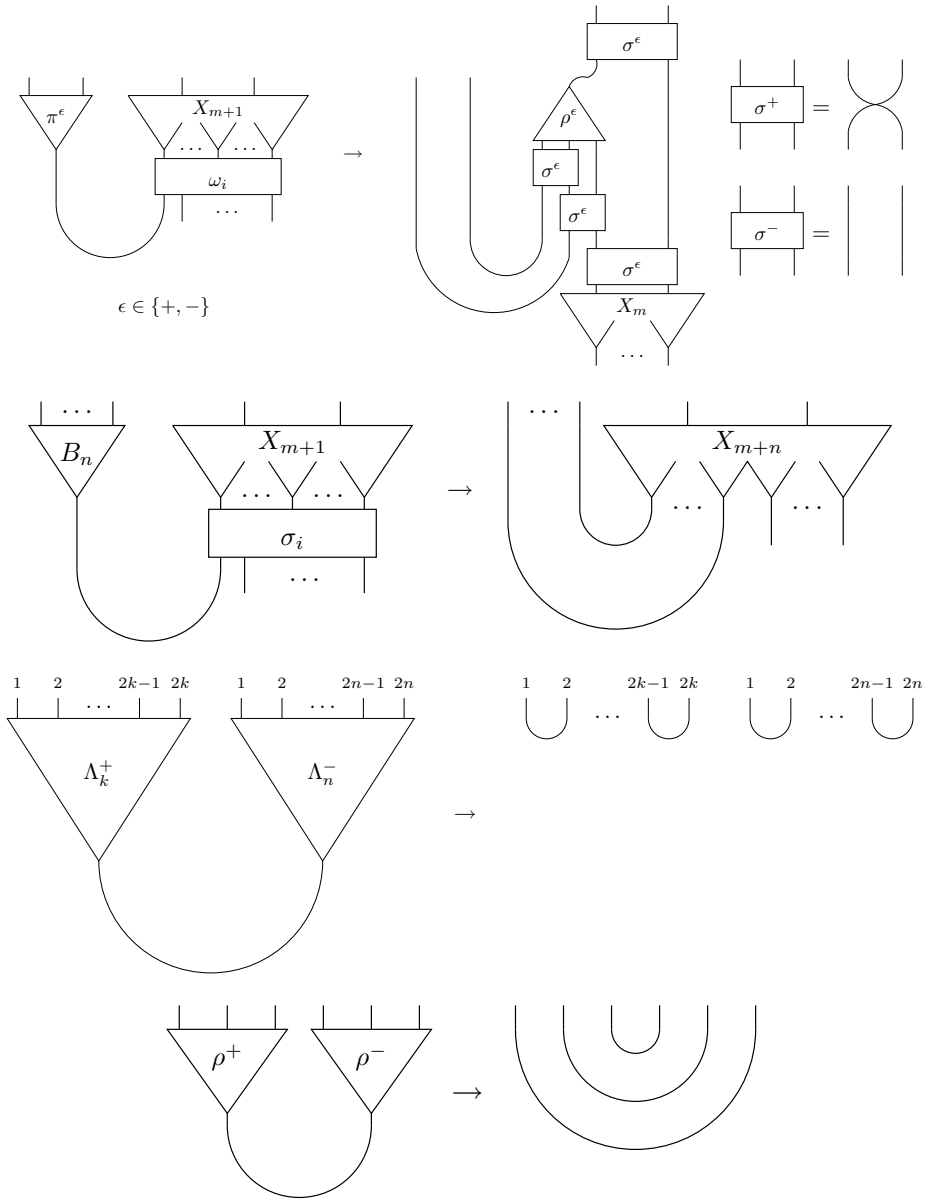


Figure 5.5: The rules of $\mathcal{F}_\infty$

labelled with names: there will be one free port labelled x for each free occurrence of x in P. In particular, the presence of a free principal port labelled by x will mean that x is the subject of a prefix, i.e., $P{\downarrow_{\overline{x}}}$ or $P{\downarrow_x}$. In our graphical representations, we will always collect principal and auxiliary free ports resp. to the bottom and to the top of the net, so if P is a process with k observables, $\langle P \rangle$ will be pictured as a net with k free ports on the bottom.

$\langle P \rangle$ might as well contain active pairs; if we need to translate $\pi.P$, we must

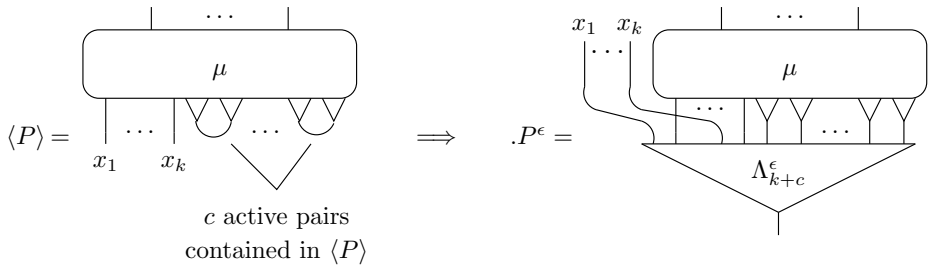
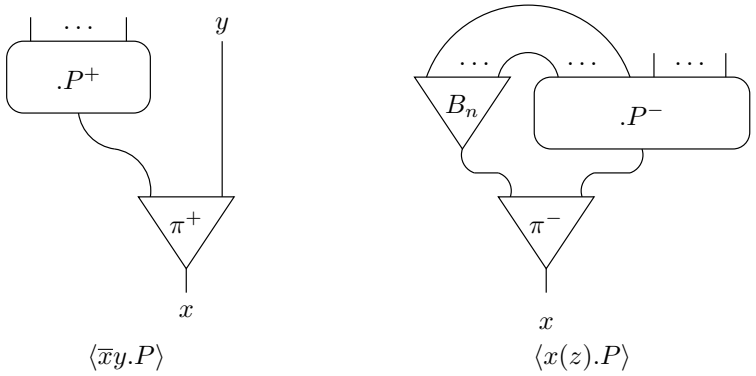


Figure 5.6: The construction of $.P^\epsilon$ from a net of the form $\langle P\rangle$.

"inhibit" such active pairs to correctly represent prefixing. To this purpose we define the nets $.P^\epsilon$ as in Fig. 5.6. An important case is $.\mathbf{0}^\epsilon$, which is just a single 0-ary Λ^ϵ_0 cell.

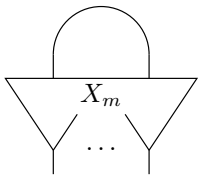
Definition 5.6 (Translation $\langle\cdot\rangle$ for $\mathrm{F}\pi$) *We define $\langle P\rangle$ by induction on P:*

- $\langle\mathbf{0}\rangle$ *is the empty net.*
- $\langle\pi.P\rangle$ *is the following net, depending on the nature of π:*



  In the encoding of the input prefix, the n free ports of $.P^-$ labelled by z are connected to the B_n cell.

- $\langle P \mid Q\rangle$ *is the net obtained by juxtaposing $\langle P\rangle$ and $\langle Q\rangle$.*
- *If $\langle P\rangle$ has m free ports labelled by z, then $\langle\boldsymbol{\nu}(z)P\rangle$ is the net obtained from $\langle P\rangle$ by connecting all such free ports to the free ports of the following net:*



  Notice that this is the only case in which active pairs may be introduced.

If $\langle P\rangle$ has a free port labelled by x which is the principal port of a π^+ (resp. π^-) cell, we write $\langle P\rangle_{\overline{x}}$ (resp. $\langle P\rangle_x$).

We have not mentioned types, but the reader can check that all the nets of Definition 5.6 are well typed. Also, the encoding is defined modulo the ordering of the connections to the ports of B_n, Λ^ϵ_n, and X_m cells, which is irrelevant.

As expected, structural congruence is captured by the encoding:

Proposition 5.3 *If $P \equiv Q$, then $\langle P\rangle = \langle Q\rangle$.*

Proof. The axioms concerning parallel composition and $\mathbf{0}$ are obvious, the ones concerning restriction are also easily checked. The details are left to the reader. □

We remark that the converse of Proposition 5.3 is false; in fact, whenever z does not appear in the prefix π, $\langle \boldsymbol{\nu}(z)\pi.P\rangle = \langle \pi.\boldsymbol{\nu}(z)P\rangle$, but the two processes are not structurally congruent; they are strong barbed congruent though. We do not known how much more of strong barbed congruence is captured by the translation, although we suspect that the one pointed out here is actually the only relevant example.

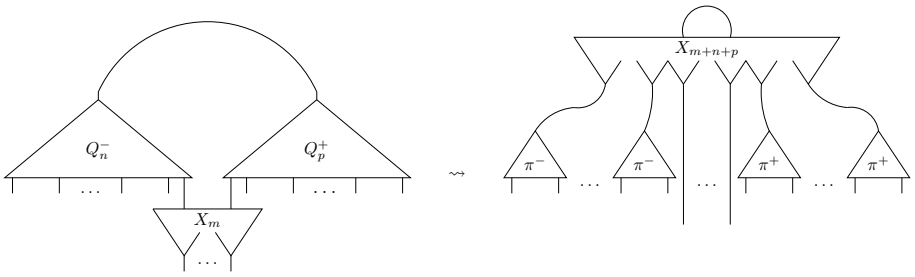
A single reduction step in the π-calculus is simulated by several interaction steps. Therefore, if μ is reduct of the translation of a process, there may be no process P such that $\langle P\rangle = \mu$. As a matter of fact, reduction commutes with the translation only modulo a *readback* procedure.

Definition 5.7 (Bureaucratic cuts) *Active pairs formed by B_n and X_m cells and by Λ^ϵ_n cells are called* bureaucratic*, and so are their respective reductions (which are resp. the second and third from the top in Fig. 5.5). We call* bureau-free *a net which contains no bureaucratic active pair.*

The following is immediate:

Lemma 5.4 *Bureaucratic reduction is terminating and (strongly) confluent; hence, any net μ has a unique associated* bureau-free *form $\mu^\flat$.*

Definition 5.8 (Readback) *Let μ be a net of $\mathcal{F}_\infty$. The* readback *of μ, noted $\widehat{\mu}$, is the net obtained by taking $\mu^\flat$ and applying, until no longer possible, the following replacement:*



where Q^ϵ_k, for $k \geq 1$, is a tree of k ρ^ϵ cells (built in the only possible way induced by the typing).

Basically, the readback procedure "undoes" the choices made in queuing up prefixes. Observe that, clearly, $\widehat{\langle P\rangle} = \langle P\rangle$ for all P.

Definition 5.9 (Synchronization step) *The $\rho^+\rho^-$ interaction rule (the last shown in Fig. 5.5) is called the* synchronization rule*, and so are rewriting steps using it. If μ, μ' are two nets of $\mathcal{F}_\infty$, we write $\mu \twoheadrightarrow \mu'$ iff $\mu \to^* \mu'$ and the reduction sequence leading from μ to μ' contains exactly one synchronization step.*

The following result is a trivial consequence of the above definitions:

Lemma 5.5 *Let P be a process, and μ a reduct of $\langle P \rangle$. If μ contains an active pair formed by a ρ^+ and a ρ^- cell, then the principal ports of the corresponding π^+ and π^- cells in $\widehat{\mu}$ are connected to two principal ports of the same X_m multicell.*

Of course synchronizations steps do not alter the confluence of bureaucratic reductions, because they involve only unicells. In other words, the real choice is done *before* the synchronization step.

Lemma 5.6 *The reduction relation consisting of bureaucratic reductions and synchronization steps is (strongly) confluent.*

We shall now prove the faithfulness of our encoding with respect to Fπ, by establishing a sort of strong barbed bisimulation between P and its translation $\langle P \rangle$. We deem this faithfulness "partial" because it is formulated in terms of fully invisible transitions (cf. Definition 5.2), contrarily to how strong barbed bisimilarity is usually formulated (cf. Definition 5.4).

Theorem 5.7 (Partial faithfulness of $\langle \cdot \rangle$) *Let P be an* Fπ *process.*

1. **Completeness:**
   (a) *If $P{\downarrow_{\mathsf{x}}}$, then $\langle P \rangle_{\mathsf{x}}$.*
   (b) *If $P \xrightarrow{\tilde{\tau}} Q$, then $\langle P \rangle \twoheadrightarrow \langle Q \rangle$.*
2. **Soundness:**
   (a) *If $\langle P \rangle_{\mathsf{x}}$, then $P{\downarrow_{\mathsf{x}}}$.*
   (b) *If $\langle P \rangle \twoheadrightarrow \mu$, then $P \xrightarrow{\tilde{\tau}} Q$ and $\widehat{\mu} = \langle Q \rangle$.*

Proof. The fact that $P{\downarrow_{\mathsf{x}}}$ iff $\langle P \rangle_{\mathsf{x}}$ for any name or coname x is trivial. To prove part 1b, one just observes that $P \xrightarrow{\tilde{\tau}} Q$ means that

$$P \equiv \boldsymbol{\nu}(z, \widetilde{w})(\overline{z}x.R_1 \mid z(y).R_2 \mid S)$$

and

$$Q \equiv \boldsymbol{\nu}(z, \widetilde{w})(R_1 \mid R_2\{x/y\} \mid S)$$

(this is the Harmony Lemma 5.2, together with Proposition 5.1), so, using Proposition 5.3, $\langle P \rangle$ contains a π^+ and a π^- cell whose principal ports are connected to two principal ports of the same X_m cell; knowing this, one easily finds a chain of 5 reductions leading to $\langle Q \rangle$, exactly one of which is a synchronization step.

Part 2b requires some more work. We need two intermediate results:

Claim 5.7.1 *Let μ_1 be a reduct of the encoding of a process, such that $\widehat{\mu_1} = \langle P \rangle$ for some process P, and such that $\mu_1 \to \mu_2$ through a synchronization step. Then, there exists a process Q such that $P \xrightarrow{\tau} Q$ and $\widehat{\mu_2} = \langle Q \rangle$.*

Proof. By hypothesis, μ_1 contains an active pair containing ρ^ϵ cells, so Lemma 5.5 applies; as a consequence, P contains an output and an input prefix acting on the same channel, and this channel is private (otherwise no X_m cell would appear in the encoding). Therefore, we can write

$$P \equiv \boldsymbol{\nu}(x, \widetilde{w})(\overline{x}y.R_1 \mid x(z).R_2 \mid S).$$

The synchronization step leading to μ_2 introduces (at most) two bureaucratic cuts; we can assume these to be the only bureaucratic cuts in μ_2, because, thanks to Lemma 5.6, if μ_1 was not bureau-free, reducing its cuts before or after the application of the synchronization step has no effect on μ_2^{b} (and thus on $\widehat{\mu_2}$). It is then just a matter of applying a few rewriting rules to check that

$$\widehat{\mu_2} = \langle \boldsymbol{\nu}(x, \widetilde{w})(R_1 \mid R_2\{y/z\} \mid S) \rangle,$$

as needed to prove our statement. □

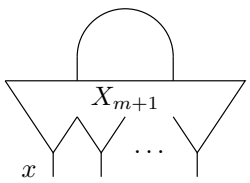
Claim 5.7.2 *Let $\mu \to \nu$ through a step different from a synchronization step. Then, $\widehat{\mu} = \widehat{\nu}$.*

Proof. If the reduction results from applying a bureaucratic rule, then μ is not bureau-free, and by Lemma 5.4 $\nu^{\mathsf{b}} = \mu^{\mathsf{b}}$, hence $\widehat{\nu} = \widehat{\mu}$. If the step is an application of a π^ϵ/X_m rule (top of Fig. 5.5), the readback "undoes" the reduction and we have again $\widehat{\nu} = \widehat{\mu}$ □

Now, by definition, $\langle P \rangle \twoheadrightarrow \mu$ means that there exist μ_1, μ_2 such that $\langle P \rangle \to^* \mu_1$ through non-synchronization steps, $\mu_1 \to \mu_2$ through a synchronization step, and $\mu_2 \to^* \mu$ again through a number of non-synchronization steps. Following the remark made just after the definition of readback, $\widehat{\langle P \rangle} = \langle P \rangle$, which by Claim 5.2 implies that $\widehat{\mu_1} = \langle P \rangle$. Now Claim 5.7.1 applies, giving us a process Q such that $P \xrightarrow{\tau} Q$ and $\widehat{\mu_2} = \langle Q \rangle$; using again Claim 5.2, we obtain $\widehat{\mu} = \langle Q \rangle$ as well. Notice that the typing discipline followed by $\mathcal{F}_\infty$ is fundamental here, because it assures us that no active pair other than those considered can arise through reduction. □

Another translation, noted $[\cdot]$, is needed if we want to account for τ transitions which are not fully invisible, i.e., which are due to synchronization on free channels. Basically, $[P]$ is a sort of "closure" of $\langle P \rangle$, i.e., $[P]$ is practically identical to $\langle \boldsymbol{\nu}(\widetilde{x})P \rangle$, where $\widetilde{x}$ are the free names of P, the only difference being that we want to remember the names we artificially bound:

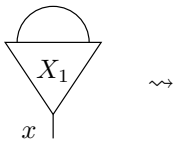
Definition 5.10 (Translation $[\cdot]$ for $\mathrm{F}\pi$) *Let x range over $\mathsf{fn}(P)$; if in $\langle P \rangle$ there are m free ports labelled by x, we define $[P]$ as the net obtained from $\langle P \rangle$ by connecting all such ports to a X_{m+1} cell, which will be left with one free port labelled by x:*



Hence, in general, $[P]$ has as many free ports as the number of free names in P. Notice that Proposition 5.3 transfers to $[\cdot]$ without any problem.

Now, the number of free ports is stable under reduction, while free names are not (some may disappear). This is why we need the following extension to the readback operation:

Definition 5.11 (Readback, extended) *Let μ be a net of $\mathcal{F}_\infty$. The* extended readback *of μ, noted $\widetilde{\mu}$, is the net obtained by taking $\widehat{\mu}$ and removing all isolated X_1 cells, i.e., applying, until no longer possible, the following replacement:*

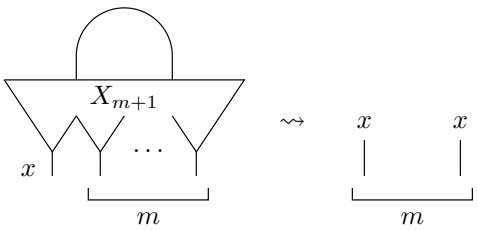


The following is an almost obvious consequence of Theorem 5.7, considering the above remark that $[P]$ is virtually identical to $\langle \boldsymbol{\nu}(\widetilde{x})P \rangle$, where $\widetilde{x} = \mathsf{fn}(P)$.

Theorem 5.8 (Full faithfullness of $[\cdot]$) *Let P be an* Fπ *process.*

1. **Completeness:** *If $P \xrightarrow{\tau} Q$, then $[P] \twoheadrightarrow \mu$ such that $\widetilde{\mu} = [Q]$.*
2. **Soundness:** *If $[P] \twoheadrightarrow \mu$, then $P \xrightarrow{\tau} Q$ such that $[Q] = \widetilde{\mu}$.*

Of course, any net of the form $[P]$ can be brought to the form $\langle P \rangle$ by simply applying a further readback operation, consisting in the removal of the artificial name restrictions:



5.3 Adding replication

The fact that mINS's are able to faithfully encode Fπ is already meaningful from the point of view of concurrent computation, but is extremely poor in terms of expressive power. In this section we shall give a stronger result by showing that the system $\mathcal{F}_\infty$ can be extended into a system $\mathcal{C}_\infty$ that encodes Cπ (cf. Sect. 5.1.4), which is Fπ augmented with replication.

The alphabet of the (infinite) typed mINS $\mathcal{C}_\infty$ is defined by adding to the alphabet of $\mathcal{F}_\infty$ the cells of Fig. 5.7, whose interactions are given by the rules of Fig. 5.8:

- There is an additional type, R, which represents *name restrictions.*

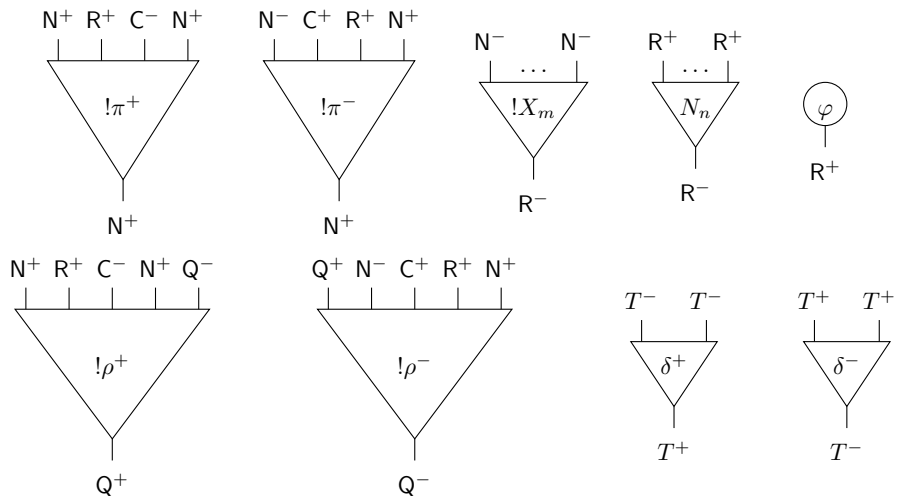


Figure 5.7: Additional cells for $\mathcal{C}_\infty$.

- $!\pi^\epsilon$ and $!\rho^\epsilon$ cells play the same role resp. as π^ϵ and ρ^ϵ cells: the first represent replicated prefixes, the second *enqueued* replicated prefixes. They carry two additional pieces of information: another name, and a restriction. The first is a *potential occurrence* of the subject of the prefix, which is needed since a replicated prefix whose subject is x potentially generates a new occurrence of x after replication. The second is a sort of pointer to the restricted names which are under the scope of the replication; these names are not usable until replication takes place.
- A cell belonging to the $!X_m$ family (m auxiliary ports, $m \geq 1$) represents a restricted name with m occurrences blocked under a replicated prefix.
- Cells belonging to the N_n family (n auxiliary ports, $n \geq 0$) "collect" $!X_m$ cells and pass them to $!\pi^\epsilon$ cells.
- The φ cell "unblocks" restricted names whenever a copy of a replicated process is activated.
- δ^ϵ cells are *duplicators*: they implement (local) replication of processes. They are *overloaded* cells, i.e., their ports can be typed with any type T, provided the typing respects the input/output specifications of Fig. 5.7.

Definition 5.12 (Translations $\langle\cdot\rangle$ and $[\cdot]$ for Cπ) *We extend the translation $\langle\cdot\rangle$ of Definition 5.6 to Cπ processes in the following way. Suppose that $.P^\epsilon$ (as always, $\epsilon \in \{+,-\}$) is a net containing n X-cells and (among others) k free ports labelled with the name z:*

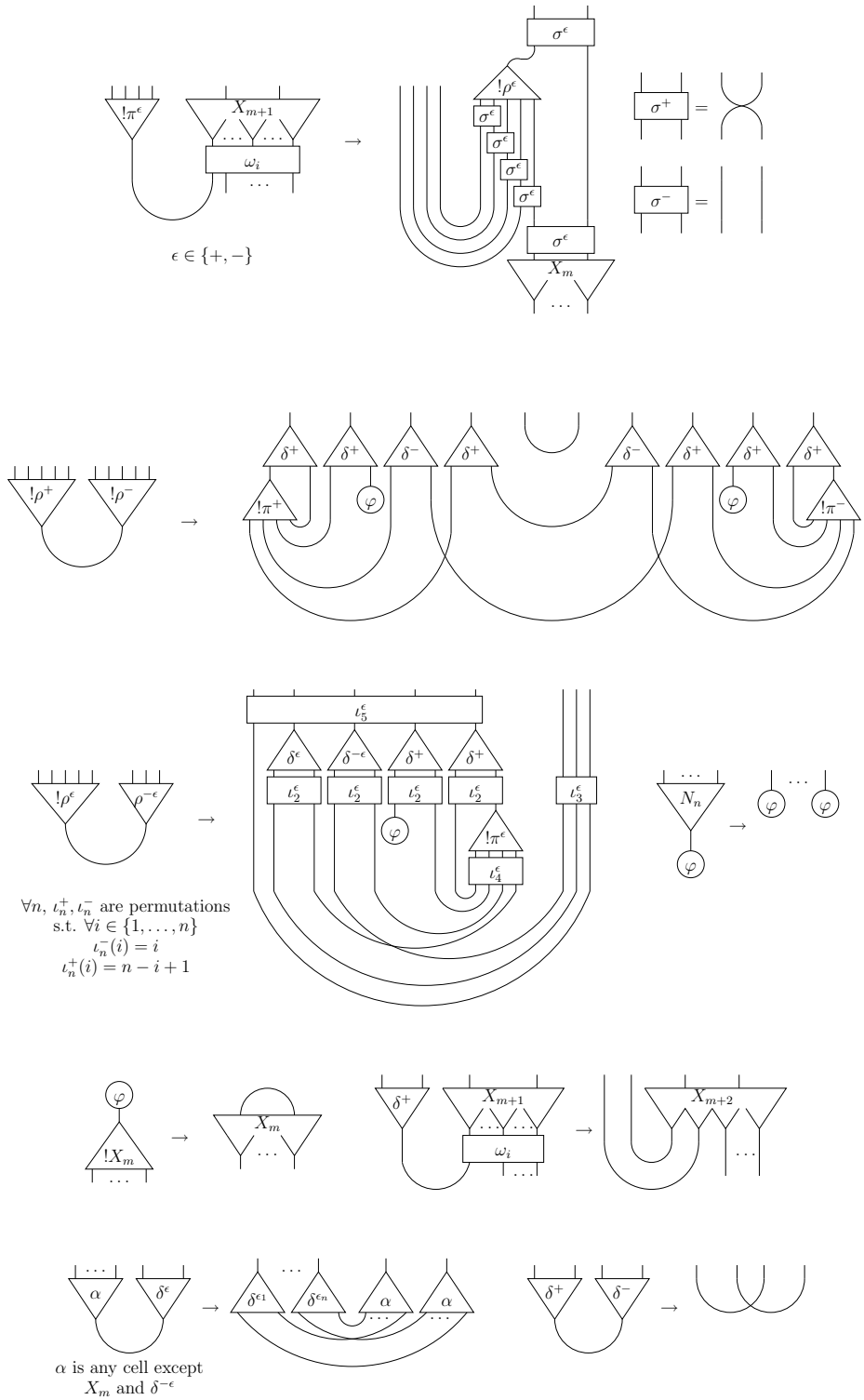
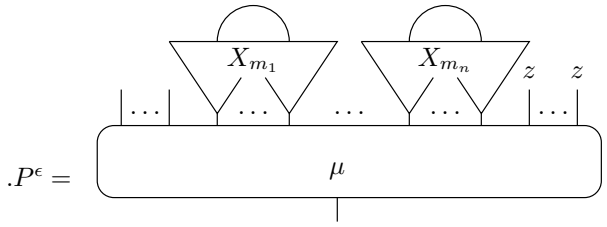
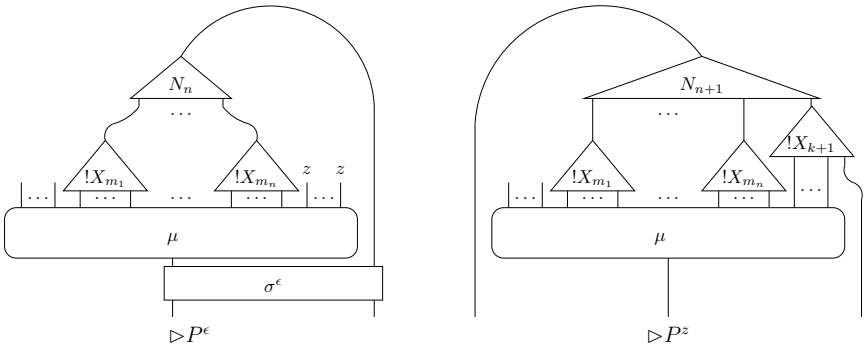


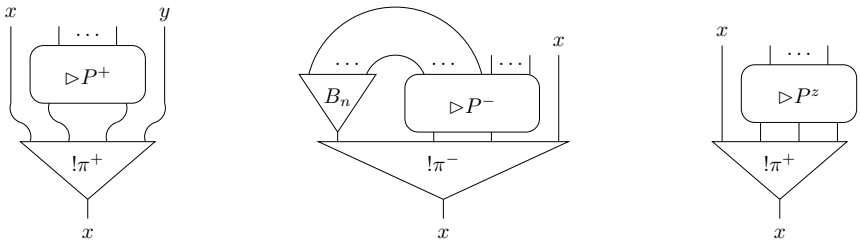
Figure 5.8: Rules for the additional cells of $\mathcal{C}_\infty$.



Then, we define $\triangleright P^{\epsilon}$ and $\triangleright P^{z}$ as follows:



where, as usual, σ^- is the identity permutation and σ^+ is the "twist" permutation. We can now define $\langle \kappa \triangleright P \rangle$:



If $\langle P \rangle$ has a free port labelled by x which is the principal port of a π^+ or $!\pi^+$ (resp. π^- or $!\pi^-$) cell, we write $\langle P \rangle \downarrow_{\overline{x}}$ (resp. $\langle P \rangle \downarrow_{x}$).

The translation $[P]$ is obtained from $\langle P \rangle$ exactly as in Definition 5.10.

Notice that it is no longer the case that each free occurrence of x in P corresponds to a free port labelled by x in $\langle P \rangle$; here, a free occurrence of x as subject of a replicated prefix generates *two* free ports. It is still the case though that the free *principal* ports of $\langle P \rangle$ are in bijection with the observables of P. We also remark that Proposition 5.3 holds trivially for both of the extended translations; this would of course be impossible if we had chosen a syntax with full replication and the the axiom $!P \equiv P \mid !P$.

Definition 5.7 can be extended to $\mathcal{C}_\infty$ by considering as bureaucratic the last 5 cuts of Fig. 5.8, i.e., all cuts involving φ and δ^{ϵ} cells; it is immediate to verify that Lemma 5.4 still holds. We can then define the readback $\widehat{\mu}$ of a $\mathcal{C}_\infty$ net μ: simply take its bureau-free form μ^{b}, and apply the substitution of Definition 5.8, where this time the queues might contain $!\rho^{\epsilon}$ cells, which need to be replaced

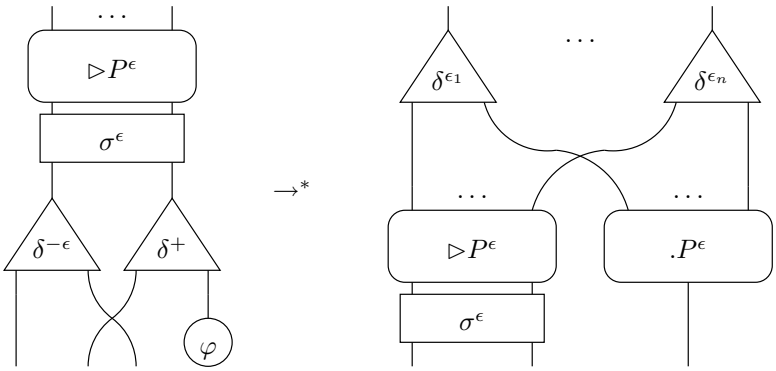
by the corresponding $!\pi^\epsilon$ cells. The extended readback $\widetilde{\cdot}$ is then defined just as in Definition 5.11.

With all the definitions extended to $\mathcal{C}_\infty$, it is not hard to prove the following:

Theorem 5.9 (Full faithfullness of the encoding $[\cdot]$ of Cπ) *Let P be a Cπ process.*

1. *$P{\downarrow_{\mathsf{x}}}$ iff $\langle P\rangle_{\mathsf{x}}$.*
2. *If $P \xrightarrow{\tau} Q$, then $[P] \twoheadrightarrow \mu$ and $\widetilde{\mu} = [Q]$.*
3. *If $[P] \twoheadrightarrow \mu$, then $P \xrightarrow{\tau} Q$ and $[Q] = \widetilde{\mu}$.*

Proof. The proof follows a similar argument to that given for Theorem 5.7. The fundamental issue concerning replication is that neither active pairs, nor X_m or δ^ϵ cells can be duplicated by δ^ϵ cells. This is why $\triangleright P^\epsilon$ and $\triangleright P^z$ are introduced: such nets are cut-free, and do not contain either X_m or δ^ϵ cells, so they can be safely duplicated. Then, φ cells extract P from $\triangleright P^\epsilon$ or $\triangleright P^z$. In other words, the reader can check that the following property holds:



and similarly for $\triangleright P^z$ nets. The other two important points are that duplication is completely bureaucratic (therefore strongly confluent), and that it stops on free channels; this assures us that the replication process does not interfere with prefix synchronization. □

Of course, the same result holds for the translation $\langle\cdot\rangle$ replacing invisible transitions with fully invisible ones.

5.4 Using finite mINS's

We have shown in Sect. 5.3 that a very expressive fragment of the π-calculus can be faithfully represented by the mINS $\mathcal{C}_\infty$. This system has an unpleasant feature though: its alphabet is *infinite*, which is a little against the idea that mINS's ought to be seen as programming languages with a finite number of primitives. Moreover, even though reductions are local, their execution time depends on the arity and co-arity of cells, and is thus not constant. In what follows, we solve both problems by showing that all of the families of cells used in $\mathcal{C}_\infty$ can be replaced by one or more cells which, when suitably composed, do the same job.

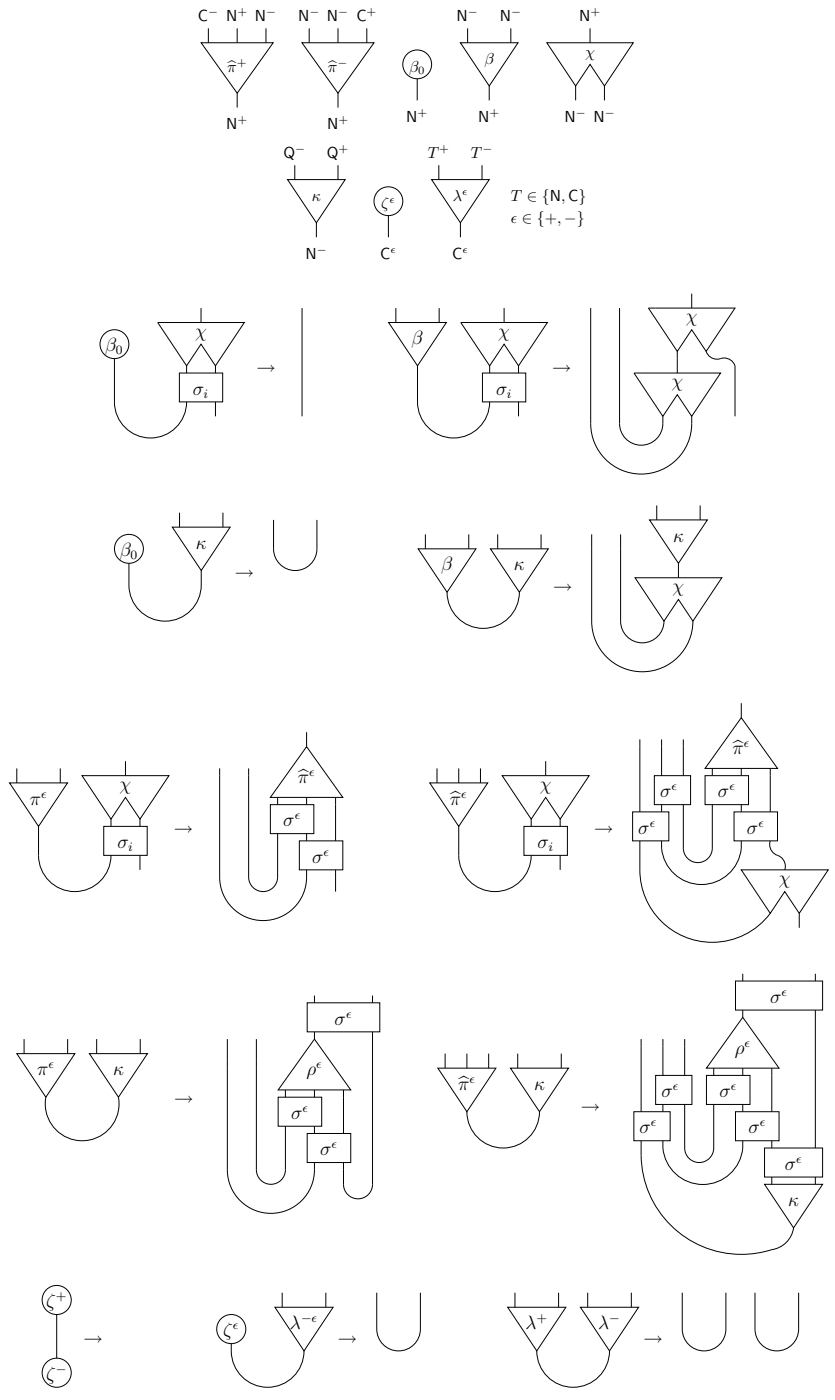
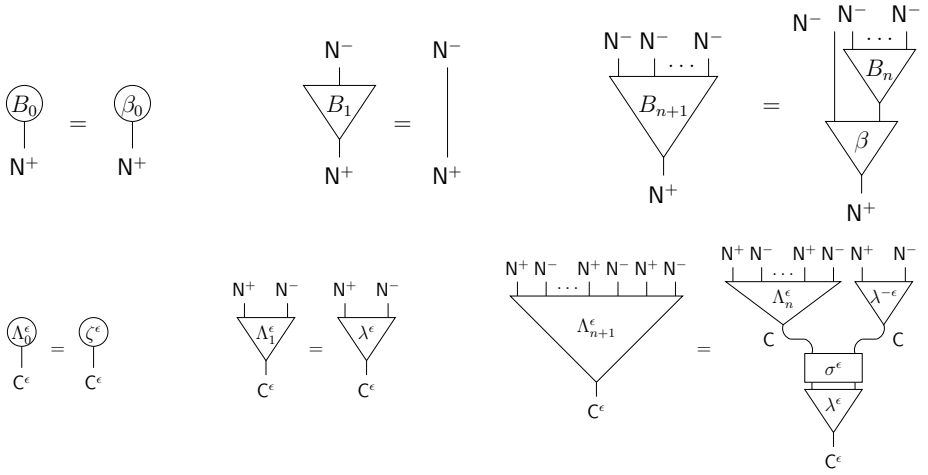
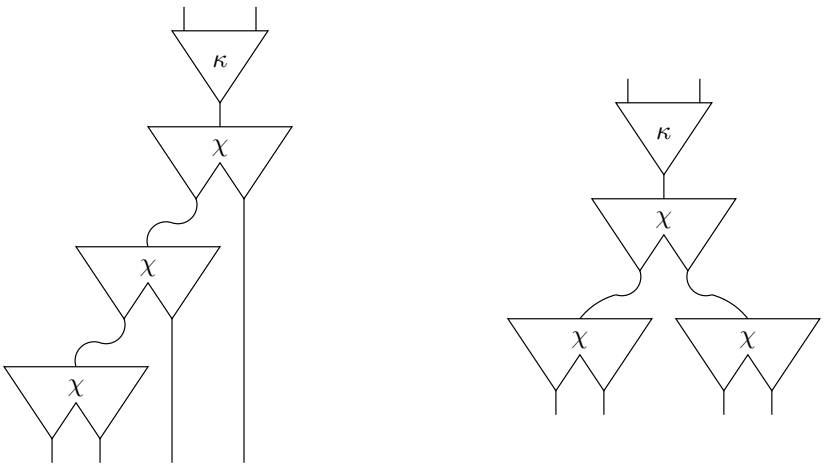


Figure 5.9: The additional cells and rules for $\mathcal{F}$.

We define $\mathcal{F}$ as the finite typed mINS whose alphabet is composed by the cells π^ϵ and ρ^ϵ of $\mathcal{F}_\infty$ plus the cells (and rules) of Fig. 5.9. The families B_n and Λ^ϵ_n of $\mathcal{F}_\infty$ can be inductively defined as follows:

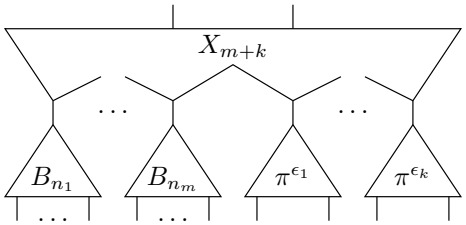


For what concerns the X_m family, we take any "tree" of $m-1$ χ cells with a κ cell "on top" to represent X_m; for example, the two configurations

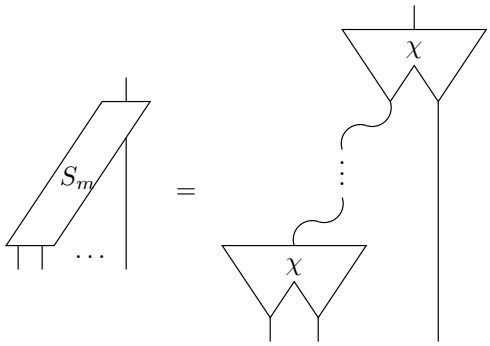


both represent the cell X_4, while X_1 is represented by a single κ cell.

We invite the reader to check that, up to this redundancy, each of the above nets faithfully simulates its corresponding $\mathcal{F}_\infty$ cell. The delicate point is obviously to verify that trees of χ cells plus a κ cell behave exactly like X_m cells. This is done by checking all possible critical pairs involving X_m cells, which are of course infinite in principle but which can be reduced to the single configuration



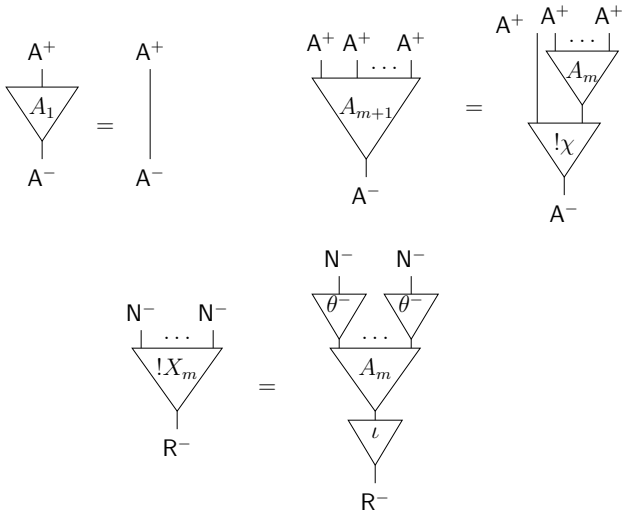
The above configuration admits as cut-free forms a $X_{n_1+\cdots+n_m}$ cell with its input and output queues filled by two trees of ρ^ϵ cells depending on the cells π^ϵ and on their order of interaction. We therefore need to check that all of the corresponding nets in $\mathcal{F}$ yield the same result. This can be done by defining "stairs" of χ cells as



where S_m contains $m-1$ χ cells. The result can be proved (by induction) on X_m cells represented using only S_m stairs; then, the result can be extended to all configurations representing X_m cells just by observing that, in general, such configurations are "stairs of stairs".

Notice that in order to correctly represent the X_m family we had to add two auxiliary cells, $\widehat{\pi}^+$ and $\widehat{\pi}^-$, which can be seen in practice as nothing but π^+ and π^- cells "navigating" through the channel until arriving to the "queuing cell" κ.

This encoding of X_m cells perfectly works in the case of $\mathcal{C}_\infty$ as well; it is sufficient to add two other auxiliary cells $!\widehat{\pi}^+$ and $!\widehat{\pi}^-$, whose interaction rules with the χ and κ cells are virtually identical to the rules $\widehat{\pi}^\epsilon/\chi$ and $\widehat{\pi}^\epsilon/\kappa$ given in Fig. 5.9, only featuring a few more wires since the cells encoding replicated prefixes have a greater arity. So one can take $\mathcal{F}$, add the cells $!\pi^\epsilon$, $!\rho^\epsilon$, φ, δ^ϵ of $\mathcal{C}_\infty$ and the cells of Fig. 5.10 (an auxiliary type A needs to be added as well) and define a finite typed mINS $\mathcal{C}$, in which the cell families of $\mathcal{C}_\infty$ can be inductively built as in the case of $\mathcal{F}_\infty$:



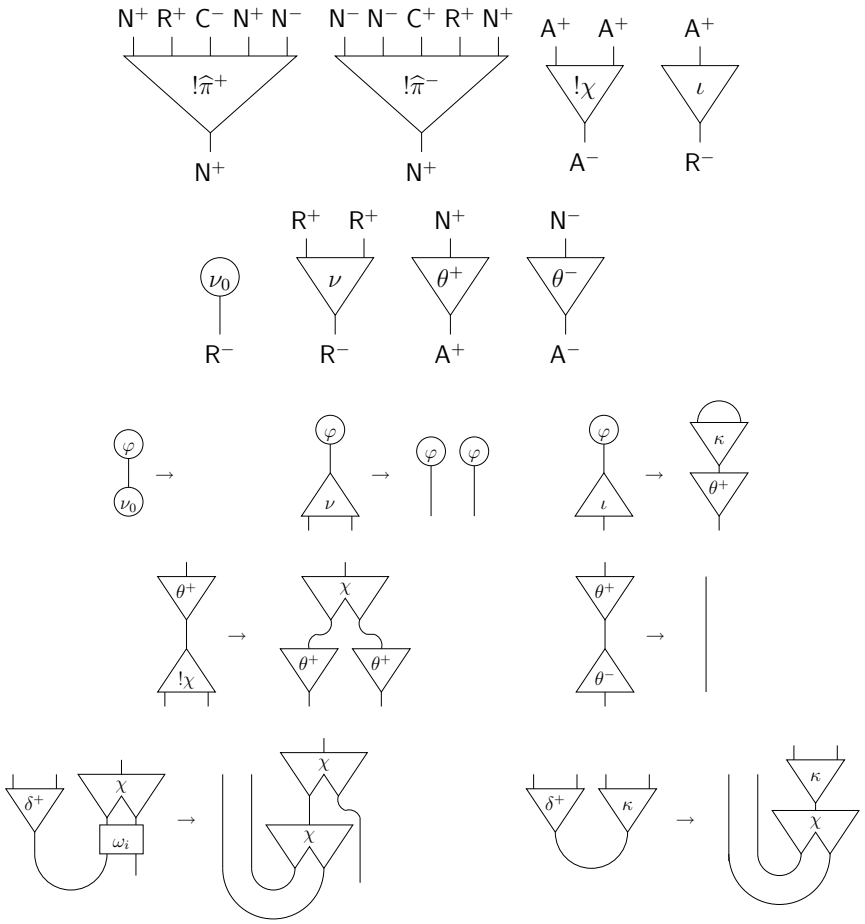
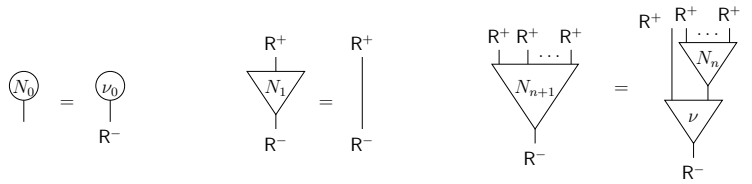


Figure 5.10: Additional cells and rules for $\mathcal{C}$ (the rules for $!\pi^\epsilon$ and $!\widehat{\pi}^\epsilon$ interacting with χ and κ have been omitted, as they are nearly identical to those for π^ϵ and $\widehat{\pi}^\epsilon$ given in Fig. 5.9).



The duplicators δ^ϵ keep... duplicating every cell as usual (see the α/δ^ϵ rule of Fig. 5.5), except for χ and κ.

So, in the end, we are left with a mINS $\mathcal{C}$ which has 5 types, 29 cells, and 48 rules (but the number of rules descends to 27 if we use templates, like we did here) which is capable of faithfully encoding the "core" π-calculus, i.e., a synchronous and extended version of Pict [PT97]. This is crucial with respect to the expressive power of the multiport combinators, because we shall see that they are only able to encode *finite* mINS's (cf. Sect. 6.4).

5.5 Encoding the full π-calculus

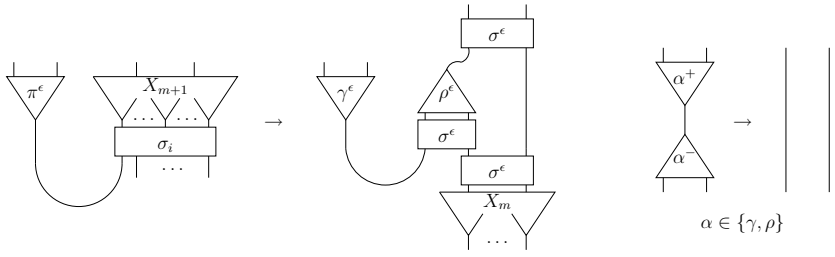
In this section we address those feature of the π-calculus which are most of the time considered "optional", i.e., that may be very useful but are not of crucial importance when modeling concurrent systems. Because of this, the exposition will be less rigorous, and only the basic ideas will be given, without detailed proofs.

5.5.1 Representing arbitrary summations

The first "optional" feature we deal with is the addition of arbitrary (finite) summations to Fπ or Cπ.

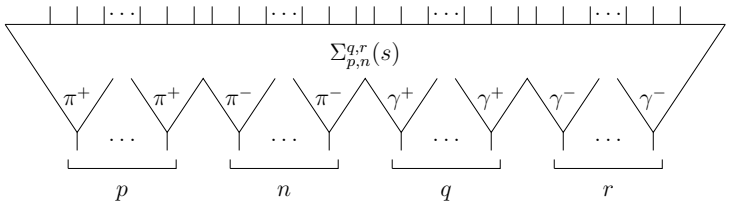
Remember from Sect. 5.1 that, when a summation interacts, all of its summands are lost, except the one which is interacting. It is quite tricky to represent this highly non-local operation within multiport interaction nets; in fact, in the presence of arbitrary summations, the encoding becomes extremely heavy and full of *ad hoc* technicalities. Here we shall only give the idea behind it, ignoring all the gory details. We shall also drop the typing, not because the systems we define cannot be typed, but because graphical notations become lighter by omitting it.

We start by remarking that the interaction between π^ϵ and X_m cells could have been defined in the following alternative way, with modified ρ^ϵ cells and two additional γ^ϵ cells:



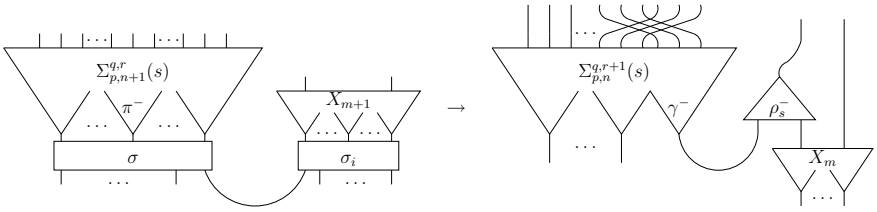
As a matter of fact, one can easily see that our "old" ρ^ϵ cells are a combination of the γ^ϵ and the "new" ρ^ϵ cells.

The basic idea to encode summations is the use of a family of multicells which somehow represent "clusters" of π^ϵ and γ^ϵ cells:



A multicell $\Sigma^{q,r}_{p,n}(s)$, where p, n, q, r, s are all non-negative integers, with p, n, q, r not all null at the same time, has $q+r+p+n$ principal ports and $3(q+r+p+n)$ auxiliary ports; the parameters p, n, r, q represent the number of principal ports behaving resp. like π^+, π^-, γ^+, and γ^- cells. The parameter s is the *summation identifier* (or *sid*) of the cell; its purpose will be explained shortly.

The interaction of a summation with a channel is defined as follows (the case of an input prefix is shown, that of an output prefix being symmetrical as always):



Notice that each principal port of a $\Sigma^{q,r}_{p,n}(s)$ cell has *three* associated auxiliary ports, and not two, as we would expect by looking at π^ϵ and γ^ϵ cells. This extra port is a "backup pointer" to the subject channel of the prefix belonging to the summation. To understand why such a backup pointer is needed, consider the following process:

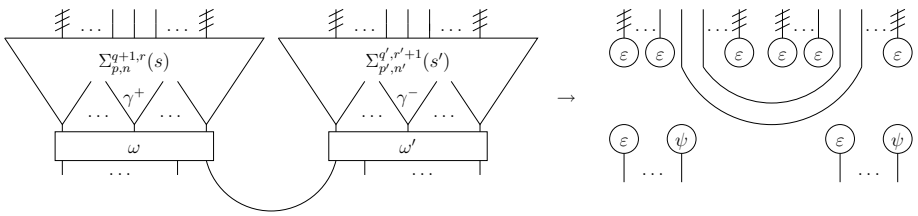
$$\overline{x}y.P + x(z).Q \mid \overline{x}a.R \mid x(w).S \ .$$

In our implementation, it might perfectly happen that all four prefixes enter the i/o queues of x before any other interaction. As a consequence, we may suppose that at some point the queuing is such that the pairs $\overline{x}a.R/x(z).Q$ and $\overline{x}y.P/x(w).S$ are bound to interact, the first with priority upon the second. But when $x(z).Q$ interacts, its companion $\overline{x}y.P$ is lost, and $x(w).S$ gets stuck. To correctly handle the situation, we can think of sending an "abort signal" to $x(w).S$; this latter prefix would then know that it has to go back to its original state, i.e., request to be re-lined up on channel x. This is the reason of the extra pointer: each prefix must remember its subject in case it receives an abort signal.

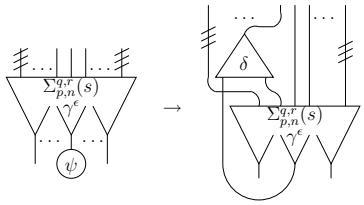
The same example brings to light another possible source of incorrectness: in fact, nobody forbids $\overline{x}y.P$ and $x(z).Q$ to enter resp. the output and input queues first, in which case we would be forcing two prefixes belonging to the same summation to interact, which is absolutely illegal (such a situation would generate the presence of a connection between two principal ports *of the same multicell*, a configuration which would be irreducible). This is why the sid is needed: this identifier, which is unique to each summation, is passed on to the

ρ^ϵ cells (which then become ρ^ϵ_s cells, i.e., they too form an infinite family), and if two cells bearing the same sid happen to interact, they just ignore each other.

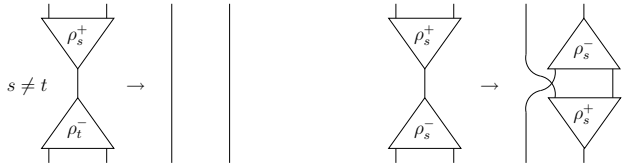
Let us see how we can implement the mechanism described through the interaction rules. The following rule is a generalization of the γ^+/γ^- rule; we have grouped triples of auxiliary ports for reasons of space:



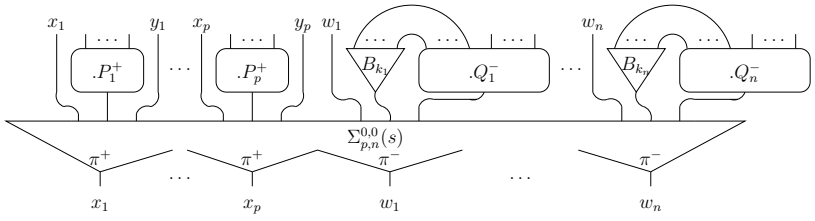
The cells ε are erasers; they erase any cell they encounter, except for X_{m+1} cells, which upon interaction with an ε cell just become X_m cells. The cells ψ are *abort* cells; they are dispatched to all principal ports marked by a γ^ϵ. When they arrive to the principal port of a summation cell, they produce the interaction below, which implements the "reset" operation:



δ is the usual duplicator cell; it is not polarized because we are not considering types. Finally, ρ^ϵ_s cells interact as follows:



We can show at last the encoding of an arbitrary summation. For notational convenience, we suppose that the summation we want to encode is split into an output and an input component, i.e., $M = \sum_{i=1}^{p} \overline{x_i} y_i.P_i + \sum_{j=1}^{n} w_j(z_j).Q_j$. Then, taken a fresh sid s, $\langle M \rangle$ is



All of the fundamental properties of the translations $\langle\cdot\rangle$ and $[\cdot]$ (basically Proposition 5.3 and Theorem 5.9) hold for this extension too.

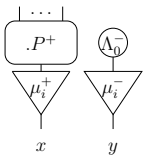
So far we have considered the addition of arbitrary summations to Fπ. The presence of replicated prefixes in Cπ needs a technical adjustment: sids must

become *pairs*, with one component set for example to zero at beginning. Then, each interaction of a summation cell with a duplicator increments this component in one of the two copies, so that a new sid is generated at each replication; we prefer to omit the details.

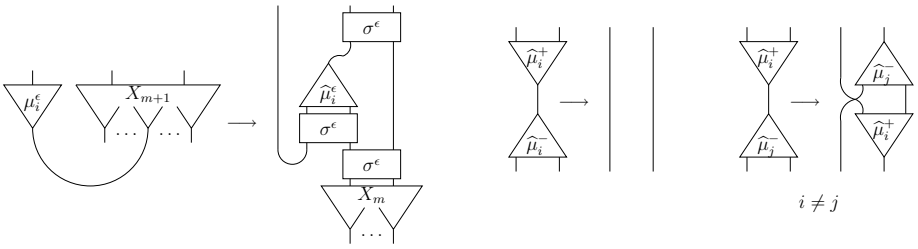
5.5.2 The internal action and match prefixes

In this section we consider other two "optional" features of the π-calculus, i.e., internal action and match prefixes. Actually, we shall only show how to encode the latter, since, as already noticed in Sect. 5.1.1, $\tau.P$ can be seen as a shorthand notation for $\boldsymbol{\nu}(z)[z = z].P$ (the two processes are in fact strong barbed congruent).

The basic idea is to use *match identifiers* (*mids*) like we did for summations. So, given a fresh mid i, we pose

$\langle [x = y].P \rangle =$ 

where the μ_i^ϵ are two new families of monocells, added together with two more families $\widehat{\mu}_i^\epsilon$, which interact as follows:



The cells $\widehat{\mu}_i^\epsilon$ also commute with any other queue cell (like ρ^ϵ), so they do not interfere with process synchronization. Replication poses no problem; it must be said however that we have lost the nice property that the free principal ports of $\langle P \rangle$ are in bijection with the observables of P.

Chapter 6

The Multiport Combinators

6.1 Translations up to

We now introduce a notion of translation from a mINS to another mINS. Our translations will be defined "up to" an equivalence relation (actually a congruence), which can be seen as a measure of its correctness:

Definition 6.1 (Translation up to) *Let $\mathcal{S} = (\Sigma, \mathcal{R})$ and $\mathcal{S}' = (\Sigma', \mathcal{R}')$ be two mINS's, $\sim$ a congruence on nets of $\mathcal{S}'$, and let Φ be an arity- and coarity-preserving function from Σ to $\wp(\Sigma')$, i.e., a function such that, for all $\alpha \in \Sigma$, α and $\Phi(\alpha)$ have the same arity and coarity; of course Φ can be trivially extended into an interface-preserving function from $\langle\Sigma\rangle$ to $\langle\Sigma'\rangle$, still denoted by Φ. We say that Φ is a* translation from $\mathcal{S}$ to $\mathcal{S}'$ up to $\sim$ *iff, for any $\alpha, \beta \in \Sigma$, and for any principal port i, j of resp. α and β such that $\mathcal{R}(\alpha^i, \beta^j)$ is defined, we have that $\Phi(\alpha^i \bowtie \beta^j) \to \mu'$ implies $\mu' \sim \Phi(\mathcal{R}(\alpha^i, \beta^j))$.*

Notice that INS translations introduced in Definition 1.5 are translations up to $\simeq_\beta$. If precongruences were allowed (which is possible), then they would actually be translations up to $\to^*$. A translation which Lafont calls *invertible* is a translation up to $=$.

The introduction of this "up to" notion of translation is necessary because of the non-deterministic nature of mINS's. In fact, the existence of a function Φ as in Definition 6.1 satisfying $\Phi(\alpha^i \bowtie \beta^j) \to^* \Phi(\mathcal{R}(\alpha^i, \beta^j))$ does not tell us that the system $\mathcal{S}'$ is able to faithfully simulate $\mathcal{S}$: a correct reduction "in isolation" does not guarantee us that, within a larger context, the encoding still behaves in a sound manner.

On the contrary, the condition of Definition 6.1 assures us that the net obtained after exactly one reduction step behaves correctly in any context, up to $\sim$. Typically, $\sim$ will be a behavioral equivalence; therefore, the stronger this equivalence, the more "convincing" the encoding will be.

The presence of the reduction step is necessary because of non-determinism: $\alpha^i \bowtie \beta^j$ in general does not behave like $\mathcal{R}(\alpha^i, \beta^j)$ in all contexts, so there is no reason for an equivalence like $\Phi(\alpha^i \bowtie \beta^j) \sim \Phi(\mathcal{R}(\alpha^i, \beta^j))$ to hold. Instead, $\alpha^i \bowtie \beta^j$ does behave like (or rather is equal to!) $\mathcal{R}(\alpha^i, \beta^j)$ after reducing it, hence the condition required above.

One may imagine to allow more than one reduction step; for example, we might require the existence of a positive integer n such that, for all μ' verify-

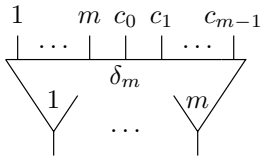
ing $\Phi(\alpha^i \bowtie \beta^j) \to^* \mu'$ in less than n steps, we still have $\mu' \sim \Phi(\alpha^i \bowtie \beta^j)$, whereas for any μ'' such that $\Phi(\alpha^i \bowtie \beta^j) \to^* \mu''$ in n or more steps, we have $\mu'' \sim \Phi(\mathcal{R}(\alpha^i, \beta^j))$. This would yield a looser, more general notion of translation; we shall not consider it on the grounds that Definition 6.1 is already enough for our purposes.

The above considerations are actually connected to the nature of *choice* in a non-deterministic setting. Since mINS's are not confluent, whenever we choose to reduce an active pair instead of another, we are potentially making an irreversible choice, i.e., we may be taking one of two non-confluent computational branches. When we encode a mINS $\mathcal{S}$ into a mINS $\mathcal{S}'$, the question is then how $\mathcal{S}'$ simulates the choices of $\mathcal{S}$. Definition 6.1 imposes that this simulation be "instantaneous": as soon as we take the first step in reducing the encoding of an active pair of $\mathcal{S}$ in $\mathcal{S}'$, we commit ourselves to the corresponding one-step choice in $\mathcal{S}$. The definition given in the paragraph above loosens this requirement, and allows the encoding a few steps of "indecision" before the commitment to a definitive choice.

6.2 The multiport combinators

The system of the multiport combinators is an infinite mINS composed by the two unicells γ and ε, of resp. arity 2 and 0, and by a family of multicells δ_m, of arity $2m$ and coarity m, for all $m > 0$.

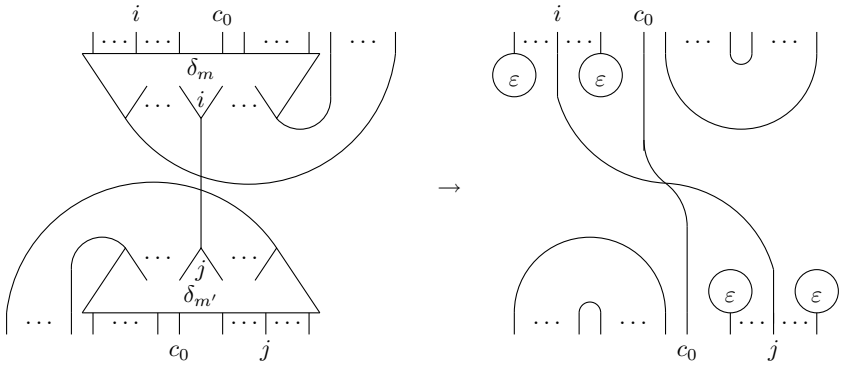
If we number from 1 to m the principal ports of δ_m, its auxiliary ports can be divided into two groups of m ports each: the first group, whose ports are also numbered from 1 to m, has one port i associated to each principal port i; the second group, has a special port c_0 and $m-1$ ports $c_1, \ldots, c_{m-1}$. Graphically, the situation can be pictured as follows:



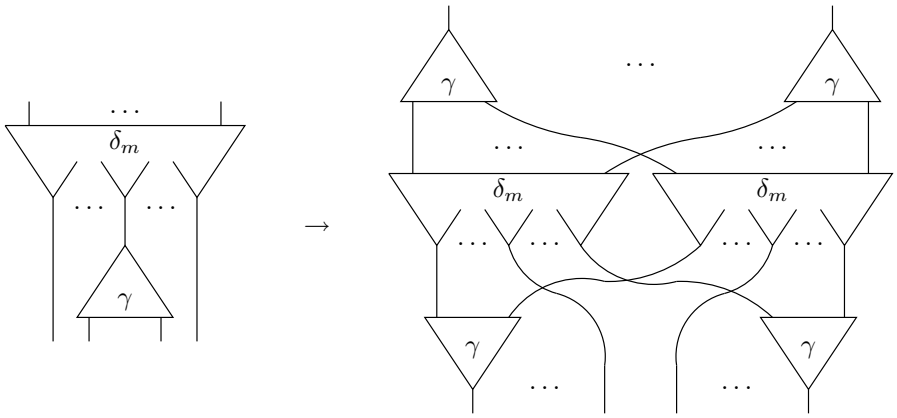
When a δ_m cell interacts with a $\delta_{m'}$, i.e., the principal port i of a δ_m cell is connected to the principal port j of a $\delta_{m'}$ cell, each of the two cells can be seen as "selecting" an auxiliary port of the first group of the other cell; moreover, each cell has access to the special port of the other. "Selecting" means that the auxiliary port i of δ_m is connected to the auxiliary port j of $\delta_{m'}$, while all other auxiliary ports of the first group of both cells receive ε cells. Then, the special port c_0 of δ_m is connected to the special port c_0 of $\delta_{m'}$.

It remains to see what becomes of the principal ports which have not interacted. This is the purpose of the ports $c_1, \ldots, c_{m-1}$. In fact, in δ_m there are exactly $m-1$ principal ports that have not interacted, numbered from 1 to $i-1$ and from $i+1$ to m. Each of these ports is connected to a c_k port: if $k < i$, then k is connected to c_k; if $k > i$, then k is connected to c_{k-1}.

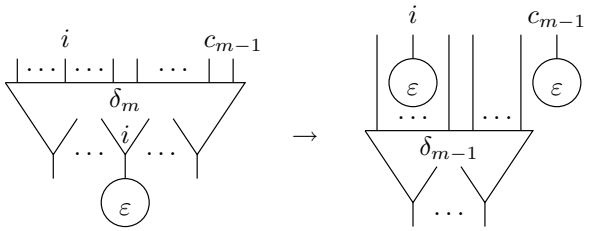
The interaction rule corresponding to the above description is the following:



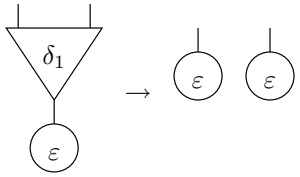
For what concerns the interaction with a γ cell, δ_m cells simply commute with it:



There is also a commutation rule for δ_m with the ε cell: if the ε cell is interacting with the principal port i of δ_m, then this cell is "passed" to the auxiliary port i of the first group, plus the "last" port of the second group, i.e., c_{m-1}. All the remaining ports are connected to a δ_{m-1} cell. The interaction just described corresponds to the the following rule:



Notice that, in case the active pair is composed of a δ_1 and an ε cell, the above rule reduces to



The rules for the $\gamma\gamma$, $\gamma\varepsilon$, and $\varepsilon\varepsilon$ interactions are defined exactly as in the interaction combinators. Therefore, the multiport combinators respect the principle which sees the interaction rules divided into two groups: the *annihilations* $\gamma\gamma$, $\delta_m\delta_{m'}$, and $\varepsilon\varepsilon$, describing what happens when two cells carrying the same symbol interact, and the *commutations* $\gamma\delta_m$, $\gamma\varepsilon$, and $\delta_m\varepsilon$, describing what happens when two cells carrying different symbols interact. The rules other than $\delta_m\delta_{m'}$, for one of $m, m' \geq 2$, are called *deterministic*.

Moreover, notice that the rules $\delta_1\delta_1$, $\gamma\delta_1$, and $\delta_1\varepsilon$ are exactly the rules $\delta\delta$, $\gamma\delta$, and $\delta\varepsilon$ of the interaction combinators. The multiport combinators are thus a conservative extension of the interaction combinators, i.e., the δ_1 cell is nothing but the "old" δ cell.

If m is a positive integer, we call *m-port combinators* the subsystem of the multiport combinators which has its alphabet restricted to $\gamma, \delta_1, \ldots, \delta_m, \varepsilon$. Each of these subsystems is finite; by the above remark, the 1-port combinators are nothing but the interaction combinators.

6.3 Behavioral equivalence for the 2-port combinators

We shall now define a notion of behavioral equivalence for the 2-port combinators. The universality of this system will in fact be proved up to this equivalence.

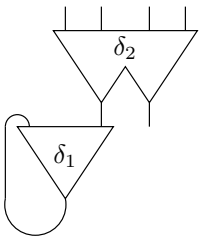
In what follows, the terms "cell", "net", and "rule" are restricted to the 2-port combinators.

Definition 6.2 (Straight path, multiport case) *Straight paths are extended to multiport nets in a natural way: a path is straight iff whenever it enters a cell through one of its principal ports, it exits trough one of its auxiliary ports, and whenever it enters through one of its auxiliary ports, it exists through one of its principal ports.*

What we would like to do now is define observable paths, and base our equivalence on them, as we did in the deterministic case. To do this, we shall be guided by the following intuition.

The fundamental property of observable paths, pointed out for example by the Separation Theorem 2.19, is that they can be "turned into wires": more precisely, whenever there exists an observable path between two free ports of a net μ with n free ports, then there exists a context C with $n+2$ free ports such that $C[\mu]$ reduces to a net having a wire connecting its two free ports. In other words, C "extracts" a wire from the observable path.

Now call μ the following net:

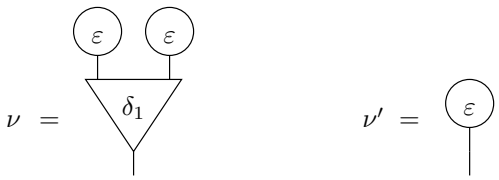


Consider the straight path starting from the free port connected to the c_1 port of the δ_2 cell and arriving to the free port connected to the second principal port of the same cell. This path does not cross any active pair, and thus would seem observable according to the definition given in the deterministic case. And yet, no context will ever be able to extract a wire from it. In fact, the only way of interacting with μ is through the principal port 2 of its δ_2 cell. If we interact with a γ cell, we only obtain, more or less, two copies of μ, so in practice we get back where we started; if we use a δ_1 cell, we end up connecting the c_1 free port to the deadlocked cell; if we use a δ_2 cell, we may in principle be able to obtain a wire, but this wire would come from the context, not from μ; if we use an ε cell, we send an ε to the c_1 free port.

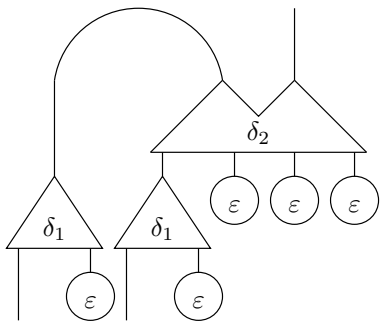
In all cases, it does not seem fair to say that the free port of μ connected to the auxiliary port c_1 of the δ_2 cell is observable. On the contrary, if both principal ports of the δ_2 cell were available for interaction, we could obtain a wire from the straight path under consideration, namely by making a δ_k cell interact with the principal port number 1. Therefore, a straight path crossing no active pair and crossing a δ_2 cell using its c_1 auxiliary port may be deemed observable only when such δ_2 cell is free to interact on both of its ports.

Considering again the above net, we have seen that a wire could indeed be extracted from it using a δ_2 cell, but we also complained that this wire came from the context rather than from μ. This phenomenon points out to us the possibility that, in the multiport combinators, δ_k cells must be bouncing, contrarily to what happens in the deterministic combinators. In fact, it would be possible to say that that the wire did come from μ, only *not* from the straight path we were considering, but from a path "bouncing off" of the second principal port of the δ_2 cell.

In order to confirm our suspects, let us consider the following two nets:



If δ_1 cells were not to be considered bouncing, then these two nets would obviously be equivalent, since they would both be blind. Yet, surprisingly, there is a context separating ν and ν' in the sense of Theorem 2.19. This context C is such that there exists a reduction of $C[\nu]$ leading to a wire, whereas all reductions starting from $C[\nu']$ converge to two ε cells (the net we called E in Theorem 2.19). The reader can check that one such context is



Therefore, δ_k cells must be considered as bouncing cells. Notice that the context used to separate ν from ν' contains an active pair. Non-determinism seems to be fundamental for discriminating between the two nets; this hints to that fact that, most probably, if a Separation Theorem holds for multiport combinators, it does not use simple tests as contexts.

The above discussion motivates the following definitions:

Definition 6.3 (Stable cell) *Let μ be a net, and α a cell of μ.*

- *We say that α is* 0*-stable iff all principal ports of α are free in μ.*
- *Let $n \geq 0$. We say that α is $n+1$-stable iff for any principal port i of α, either i is free or it is connected to an n-stable cell.*

We say that a cell α of μ is stable *iff it is n-stable for some n.*

Definition 6.4 (Residue) *Let $\mu \to \mu'$ through the reduction of an active pair $\beta_1^i \bowtie \beta_2^j$. Since interaction rules are completely local, each cell α' of μ', with the exception of those of $\mathcal{R}(\beta_1^i, \beta_2^j)$, "comes from" a unique ("the same") cell α of μ. We then say that α' is the* immediate residue *of α in μ'.*

If we have $\mu \to \mu_1 \to \cdots \to \mu_n \to \mu'$, and if $\alpha, \alpha_1, \ldots, \alpha_n, \alpha'$ are cells of resp. $\mu, \mu_1, \ldots, \mu_n, \mu'$ such that, α_1 is the immediate residue of α, α' is the immediate residue of α_n, and for all $2 \leq i \leq n$, α_i is the immediate residue of α_{i-1}, then we say that α' is the residue *of α in μ'.*

Definition 6.5 (Persistent cell) *Let μ be a net, and α a cell of μ. We say that α is* persistent *iff, for all μ' such that $\mu \to^* \mu'$, α has a residue in μ'.*

We observe that stability implies persistence, while the opposite is not true. The typical case is that of a configuration like the net μ above, in which both cells are persistent (μ admits no reduction), but none is stable.

Definition 6.6 (δ-final paths, observable δ-crossings) *A straight path is* δ-final *iff it ends into a principal port of a δ_k cell, and the port it previously crosses is* not *an auxiliary port of the same δ_k cell.*

If a straight path ϕ crosses a δ_k cell, we say that it contains a δ-crossing. Let α be the δ_k cell crossed by a δ-crossing of ϕ; we say that such a δ-crossing is observable *iff one of the following holds:*

- *ϕ crosses α (in any direction) using one of its principal ports and the c_0 auxiliary port;*
- *α is persistent, and ϕ crosses it (in any direction) using principal port i and the corresponding auxiliary port i;*
- *α is a stable δ_2 cell, and ϕ crosses it (in any direction) using one of its principal ports and the c_1 auxiliary port.*

In the following, we assume that the free ports of all nets are numbered by positive integers, so that whenever two nets have the same number of free ports, it makes sense to speak of port i for both nets (of course as long as $i \leq n$, where n is the number of free ports of the two nets).

Definition 6.7 (Observable path) *Let μ be a net, and i, j two (not necessarily distinct) free ports of μ. We say that there is an* observable path *from i to j iff one of the following holds:*

1. *$i = j$, and there is a δ-final straight path starting from i which crosses no active pair;*
2. *there is a (non-empty) straight path ϕ from i to j crossing no active pair and such that all δ-crossings of ϕ are observable.*

Definition 6.8 (Observability predicates) *Let μ be a net, and i a free port of μ. We say that i is* immediately observable *in μ, and we write $\mu{\downarrow_i}$, iff there is an observable path starting from i in μ. We say that i is* observable *in μ, and we write $\mu{\Downarrow_i}$, iff $\mu \to^* \mu'{\downarrow_i}$. If for all μ' such that $\mu \to^* \mu'$ the free port i is not immediately observable in μ', then we say that i is* blind*, and we write $\mu{\Uparrow_i}$.*

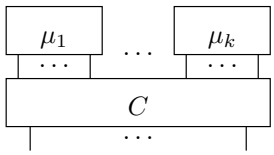
Definition 6.9 (Barbed bisimulation) *Let $\mathcal{B}$ be a relation on nets, such that if $(\mu, \nu) \in \mathcal{B}$, then μ and ν have the same number of free ports. We say that $\mathcal{B}$ is a* barbed bisimulation *iff, whenever $(\mu, \nu) \in \mathcal{B}$, we have*

1. *if $\mu{\downarrow_i}$, then $\nu{\Downarrow_i}$;*
2. *if $\mu \to \mu'$, then $\nu \to^* \nu'$ such that $(\mu', \nu') \in \mathcal{B}$;*
3. *if $\nu{\downarrow_i}$, then $\mu{\Downarrow_i}$;*
4. *if $\nu \to \nu'$, then $\mu \to^* \mu'$ such that $(\mu', \nu') \in \mathcal{B}$.*

The usual properties of bisimulations, together with all of the results proved in Sect. 2.2, hold for barbed bisimulations too. In particular, the technique of bisimulations up to reflexive-transitivity (Sect. 2.2.2) can also be applied to barbed bisimulations.

We now define a notion of multi-hole context, which trivially extends (and includes) the usual notion of context (cf. Definition 2.6).

Definition 6.10 (Multi-hole context) *A k-*hole context *for nets with n free ports is a net C with at least $kn + 1$ free ports, such that kn free ports are distinguished, and partitioned into k groups of size n each. Groups are numbered from 1 to k, and each member of a group is labelled with a couple (p, q), where p is the number of the group of membership and q is an integer from 1 to n. If $\mu_1, \ldots, \mu_k$ are k nets with n free ports, we denote by $C[\mu_1, \ldots, \mu_k]$ the net obtained by plugging each free port q of each μ_p to the free port of C labelled by (p, q). Graphically, we have*



where we have left the labelling implicit in the picture.

Definition 6.11 (Barbed congruence) *Let μ, ν be two nets with the same number of free ports. We say that μ and ν are* barbed congruent*, and we write $\mu \cong \nu$, iff, for any 1-hole context C, $C[\mu] \mathrel{\dot{\approx}} C[\nu]$.*

Barbed congruence is the multiport analogue of the congruence generated by the portwise bisimilarity briefly introduced in Sect. 2.2.3. In the interaction combinators (symmetric or not), one can prove that portwise observational equivalence coincides with $\simeq$; in the multiport combinators we are not able to prove this, so we chose to work with the stronger equivalence.

Let us show a first result concerning barbed congruence.

Lemma 6.1 (Interaction combinator reductions) *Let $\mu \to \nu$ by means of an interaction combinator reduction, i.e., a deterministic reduction using a rule among $\gamma\gamma$, $\gamma\delta_1$, $\gamma\varepsilon$, $\delta_1\delta_1$, $\delta_1\varepsilon$, and $\varepsilon\varepsilon$. Then, $\mu \cong \nu$.*

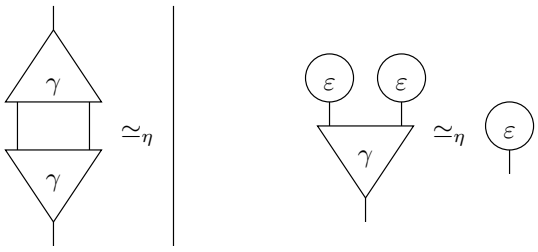
Proof. Let α, β range over the set $\{\gamma, \delta_1, \varepsilon\}$. It is enough to show that the set

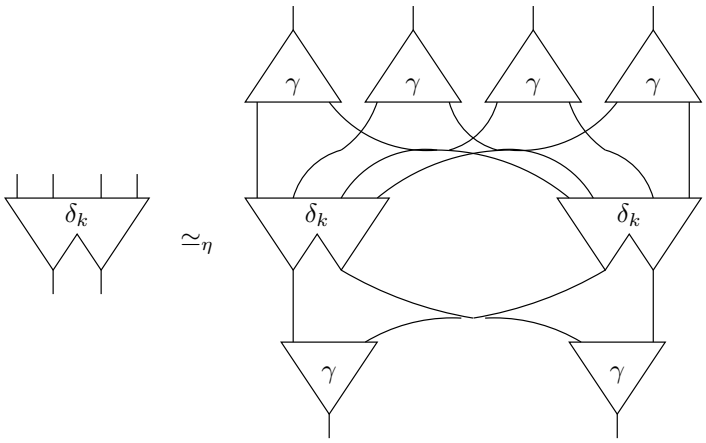
$$\mathcal{B} = \{(C[\alpha \bowtie \beta], C[\mathcal{R}(\alpha, \beta)]);\ \forall C \text{ 1-hole context}\} \cup =$$

is a barbed bisimulation. To prove this, take $(\mu, \nu) \in \mathcal{B}$. The identity relation is a barbed bisimulation, so if $\mu = \nu$ we have nothing to check. If $\mu \neq \nu$, it means that $\mu = C[\alpha \bowtie \beta]$ and $\nu = C[\mathcal{R}(\alpha, \beta)]$ for some context C. The fact that $\mu\Downarrow_i$ iff $\nu\Downarrow_i$ for all i is an immediate consequence of the form of the reduction rules under consideration. Suppose then that $\mu \to \mu'$. If the reduction takes place inside C, then $\mu' = C'[\alpha \bowtie \beta]$ for a C' such that $C \to C'$. But then $\nu \to C'[\mathcal{R}(\alpha, \beta)] = \nu'$, and by definition $(\mu', \nu') \in \mathcal{B}$. The only other possibility in case $\mu \to \mu'$ is that the active pair $\alpha \bowtie \beta$ is reduced; but then $\mu' = \nu$, and we are done since $\mathcal{B}$ contains equality. The case in which $\nu \to \nu'$ is even simpler: by definition $\mu \to \nu \to \nu'$, so once again we use the fact that equality is contained in $\mathcal{B}$. □

We remark that the key reason for the above statement to hold is that an active pair composed only by unicells cannot interact with any context, and the only way of observing it is reducing it.

Definition 6.12 (η-equivalence) *η-equivalence, denoted by $\simeq_\eta$, is the relation generated by the reflexive, symmetric, transitive, and contextual closure of the following equations:*





In the last equation, $k \in \{1, 2\}$.

The following lemma assures us that the remaining deterministic reductions also yield barbed congruent nets:

Lemma 6.2 (Deterministic reduction) *Let* $\mu \to \nu$ *by means of a deterministic reduction step. Then,* $\mu \cong \nu$.

Proof. Let μ_0 range over deterministic active pairs, and ν_0 range over their respective reducts. Set

$$\mathcal{B}_0 = \{(C[\mu_0, \ldots, \mu_0]], C[\nu_0, \ldots, \nu_0]);\ \forall \mu_0, \nu_0,\ \forall C \text{ multi-hole context}\}.$$

Now pose $\mathcal{B} = \mathcal{B}_0 \cup \simeq_\eta \cup \dot{\approx}$. We claim that $\mathcal{B}$ is a barbed bisimulation up to reflexive-transitivity, which is enough to prove the lemma.

The proof is extremely long, as it is made up of a rather conspicuous number of cases to be checked. We only examine a few of them, leaving the others to the interested reader.

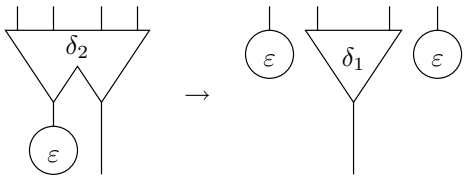
By definition, we need to take a generic $(\mu, \nu) \in \mathcal{B}$ and prove:

(1) $\mu\downarrow_i$ implies $\nu\Downarrow_i$;

(2) $\mu \to \mu'$ implies $\nu \to^* \nu'$ such that $(\mu', \nu') \in \mathcal{B}^*$,

plus the symmetrical statements (3) and (4) where the rôles of μ and ν are exchanged.

Now, if $(\mu, \nu) \in \mathcal{B}$, then we can assume that either $(\mu, \nu) \in \mathcal{B}_0$ or $\mu \simeq_\eta \nu$; in fact, in case $\mu \dot{\approx} \nu$, there is nothing to prove. Let us start with the former case. Thanks to Lemma 6.1, we only need to bother with the cases in which $\mu \to \nu$ through a $\gamma\delta_2$ or $\varepsilon\delta_2$ rule, since otherwise $\mu \dot{\approx} \nu$, and there is nothing to prove. We shall analyze to some depth the latter rule, leaving the other to the reader. The rule under examination is



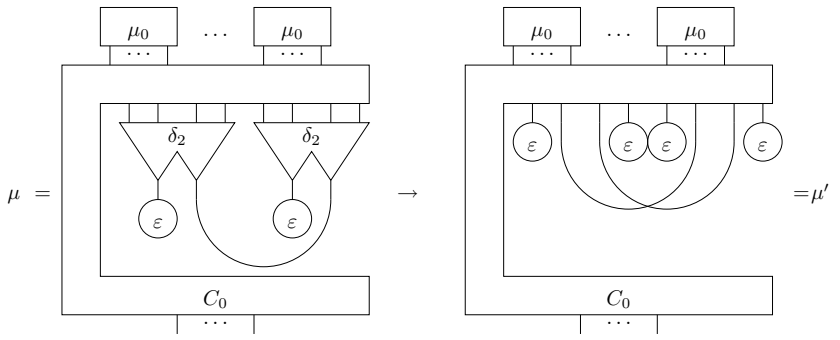
so μ_0 and ν_0 denote resp. the left and right members of the above rule, and $\mu = C[\mu_0, \ldots, \mu_0]$, $\nu = C[\nu_0, \ldots, \nu_0]$ for some multi-hole context C. Actually, there is also the case in which the ε cell is connected to the second principal port of the δ_2 cell; but this case is perfectly symmetrical, and the arguments we shall go through in the sequel can be used without problems to handle it as well (this is true also for the $\gamma\delta_2$ rule).

Let us first check that μ_0 and ν_0 (surrounded by any context) are equivalent with respect to observability, i.e., points (1) and (3) hold. Observe that the δ_2 cell in μ_0 is not persistent; therefore, the only case to be checked is that in which $\mu\downarrow_i$ because of a δ-final straight path ending into the second principal port of δ_2. This path is present in ν as well, so no problem. Conversely, there is nothing to check since whenever $\nu\downarrow_i$, we always have a reduction (the one under study!) such that $\mu \to \nu\downarrow_i$, so $\mu\Downarrow_i$ by definition.

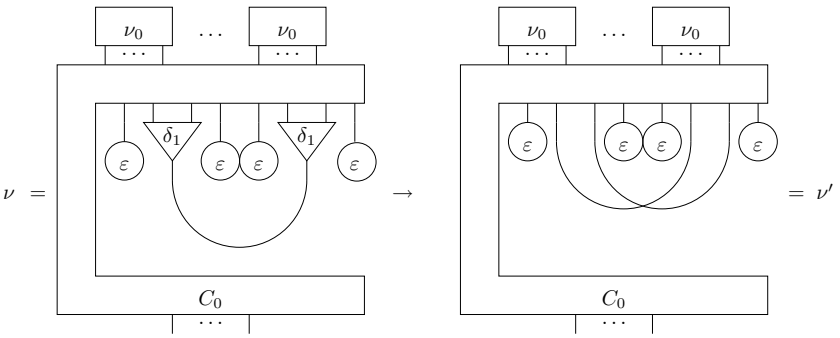
We now turn to showing point (2). Let $\mu \to \mu'$. If the reduction does not touch any copy of μ_0, ν can obviously simulate it. Otherwise, one of the cells of one of the copies of μ_0 has interacted. The first possibility is that one of the copies of μ_0 becomes ν_0 in μ', i.e.,

$$\mu = C[\mu_0, \ldots, \mu_0, \ldots, \mu_0] \to C[\mu_0, \ldots, \nu_0, \ldots, \mu_0] = \mu'.$$

By taking the occurrence of ν_0 in μ' as part of the context, we can always say that $\mu' = C'[\mu_0, \ldots, \mu_0]$; but then $\nu = C'[\nu_0, \ldots, \nu_0]$, so either $\mu' = \nu$ (in case C is a 1-hole context), and we are done since equality is contained in $\dot{\approx}$ (and thus in $\mathcal{B}^*$), or $(\mu', \nu) \in \mathcal{B}_0$ by definition. The second possibility is that a δ_2 cell in one of the copies of μ_0 interacts through its second principal port with a cell of C, or with another δ_2 of another copy of μ_0. Let us first analyze this latter case; we have

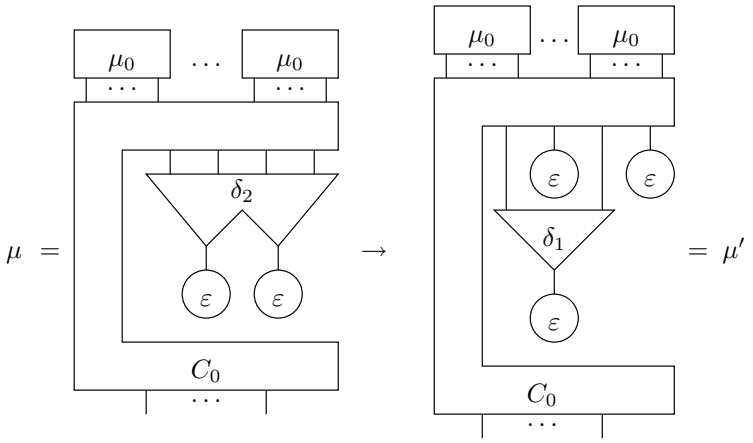


and

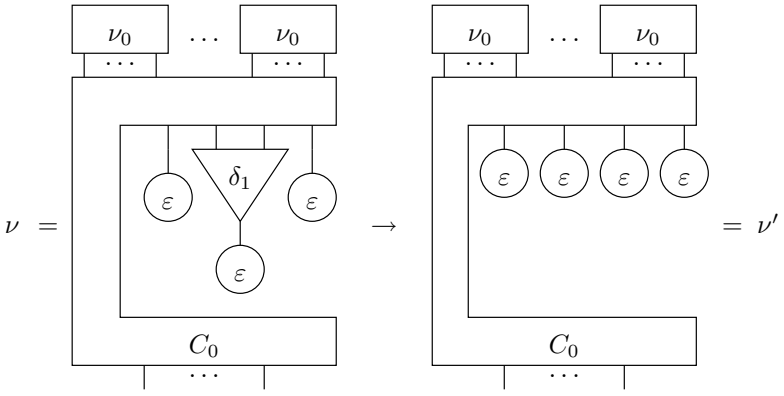


Now, as before, we can take the common part of μ' and ν', i.e., everything except the copies of μ_0 and ν_0 still remaining in resp. μ' and ν', and consider it as a context: $\mu' = C'[\mu_0, \ldots, \mu_0]$ and $\nu' = C'[\nu_0, \ldots, \nu_0]$. Once again, either $\mu' = \nu'$, or $(\mu', \nu') \in \mathcal{B}$ by definition.

We can now look at the cases in which a δ_2 cell in one of the copies of μ_0 interacts with a cell α in C. If $\alpha = \delta_k$, arguments completely analogous to the ones above work just fine. If $\alpha = \varepsilon$, we have

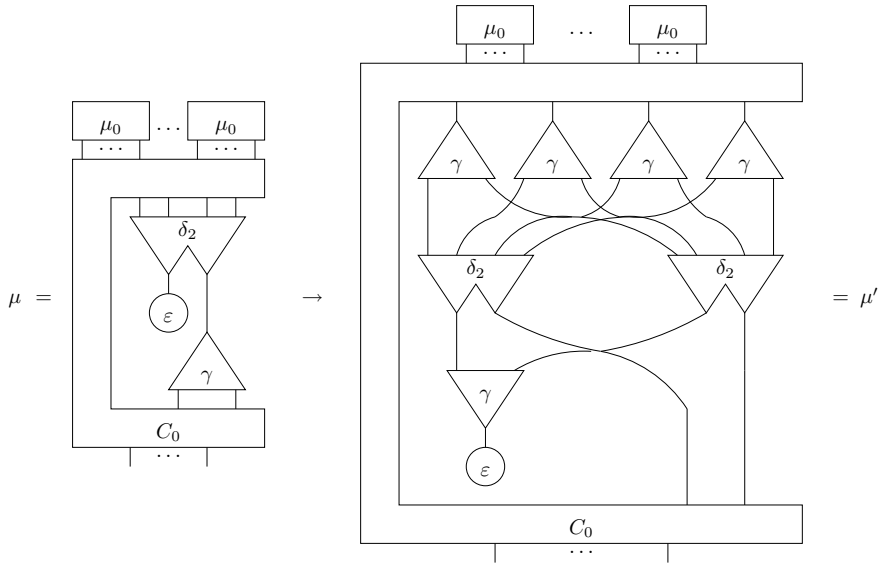


If we take ν, we can reduce it as follows:

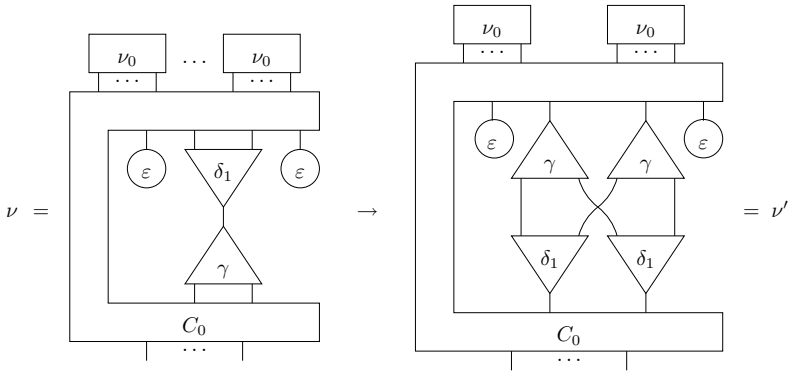


Now call C' the context made up of everything in μ' except the copies of μ_0 still left after the reduction, i.e., $\mu' = C'[\mu_0, \ldots, \mu_0]$. By Lemma 6.1, the first and third ε cells from the left shown in the picture of ν' can be transformed, modulo barbed bisimilarity, into an active pair $\delta_1 \bowtie \varepsilon$; in other words, we have $\nu' \mathrel{\dot{\approx}} C'[\nu_0, \ldots, \nu_0] = \nu''$. But barbed bisimilarity is included in $\mathcal{B}$, and $(\mu', \nu'') \in \mathcal{B}$ by definition; therefore, we can conclude by symmetry of $\dot{\approx}$ and by transitivity of $\mathcal{B}^*$ that $(\mu', \nu') \in \mathcal{B}^*$, as desired.

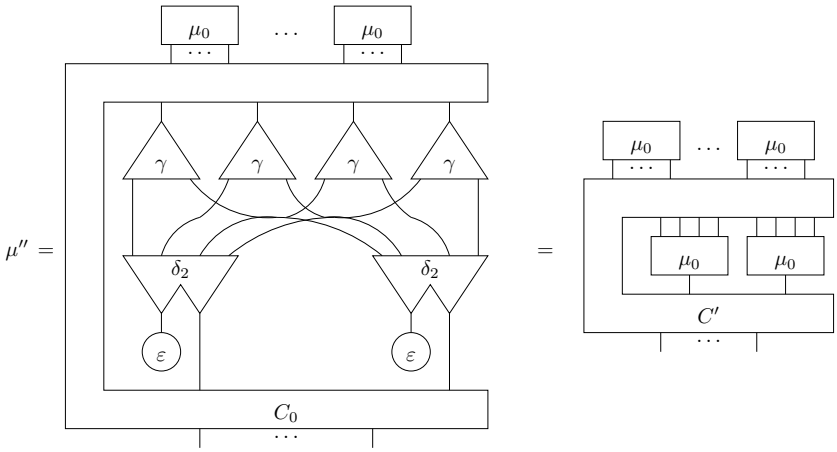
It remains the case $\alpha = \gamma$. We have



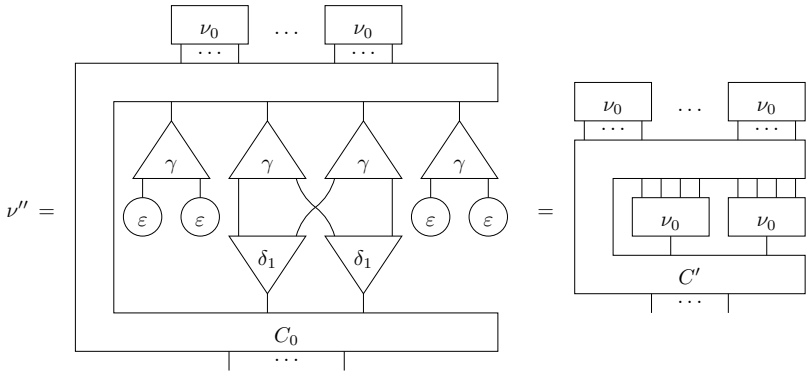
ν admits the following reduction:



By Lemma 6.1, $\mu' \dot{\approx} \mu''$, where

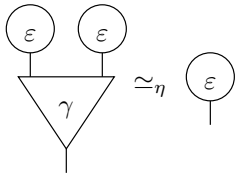


On the other hand, we have $\nu' \simeq_\eta \nu''$, where

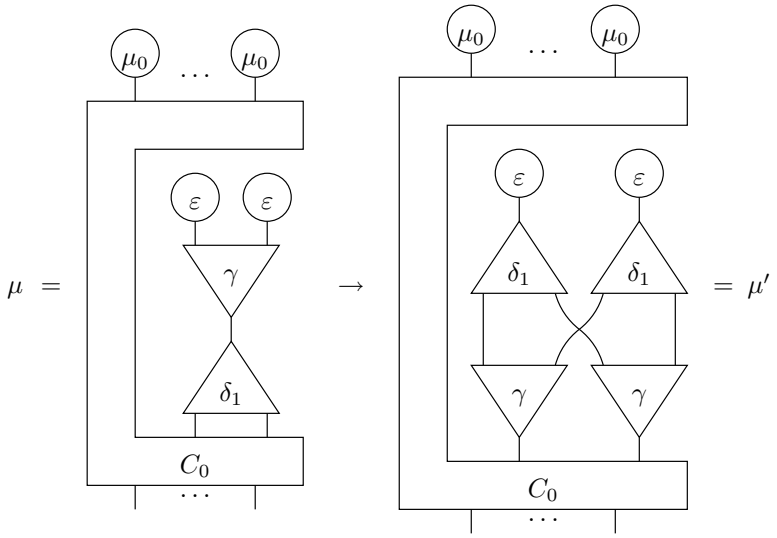


Now, by definition, $(\mu'', \nu'') \in \mathcal{B}$; additionally, since $\dot{\approx}$ and $\simeq_\eta$ are both included in $\mathcal{B}$, and since η-equivalence is symmetric, we also have $(\mu', \mu''), (\nu'', \nu') \in \mathcal{B}$. Then, by transitivity, $(\mu', \nu') \in \mathcal{B}^*$.

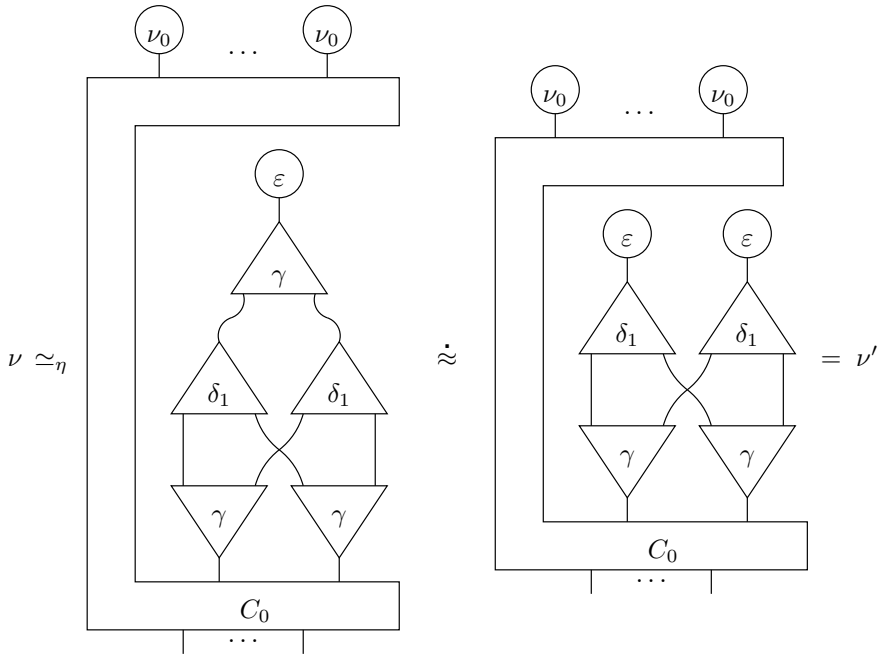
This shows that property (2) holds if $(\mu, \nu) \in \mathcal{B}_0$; property (4) is shown by similar arguments. We omit the details, and pass directly to the analysis of a few cases in which $\mu \simeq_\eta \nu$. We shall start by verifying in some depth the case of the equation



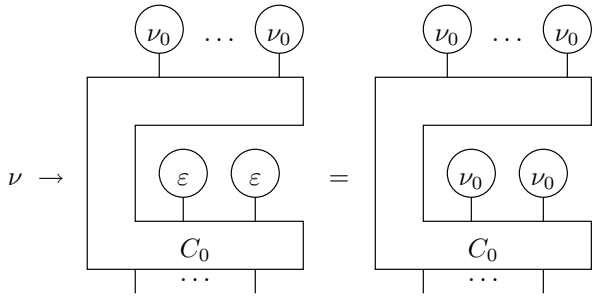
As above, we call μ_0 the left member and ν_0 the right member, so that $\mu = C[\mu_0, \ldots, \mu_0]$ and $\nu = C[\nu_0, \ldots \nu_0]$ for some multi-hole context C. We remark that there can be no observable path using any of the two configurations, so there is nothing to check as far as points (1) and (3) are concerned. Then, suppose $\mu \to \mu'$ (or $\nu \to \nu'$ to check property (4)). As before, if $\mu = C[\mu_0, \ldots, \mu_0] \to C'[\mu_0, \ldots, \mu_0]$, then ν has no problem simulating μ, and vice versa. So we need to take care of the cases in which one of the copies of μ_0 interacts with the context, or with one of the other copies (and similarly, for property (4), replacing μ_0 with ν_0). The case of the interaction between two copies of μ_0 or ν_0 is easy. If one of the copies of μ_0 interacts with a cell α of C, the only interesting cases are those in which $\alpha = \delta_1$ and $\alpha = \delta_2$. In the first case, we have



Now, we take ν and we observe that, using Lemma 6.1, we can write

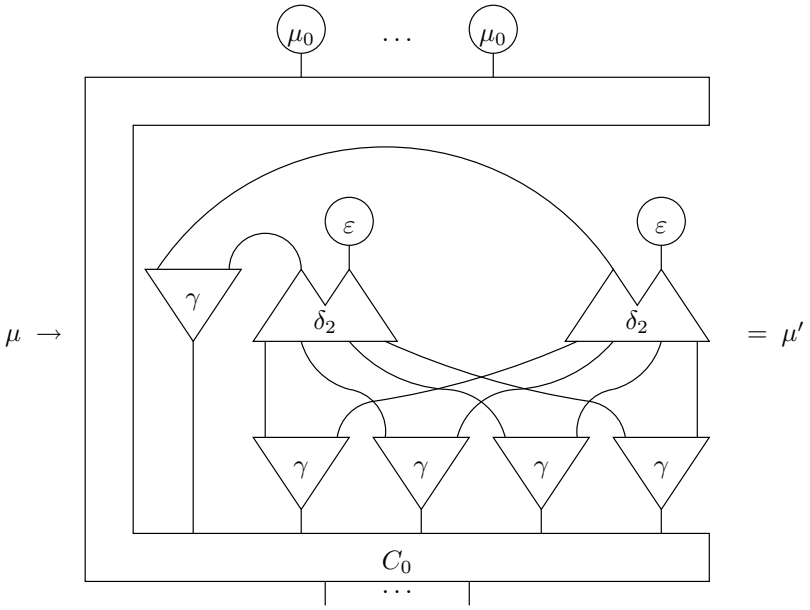


By the fact that both $\simeq_\eta$ and $\dot{\approx}$ are symmetric and included in $\mathcal{B}^*$, and by transitivity, we obtain $(\nu', \nu) \in \mathcal{B}^*$. But $\mu' = C'[\mu_0, \ldots, \nu_0]$ and $\nu' = C'[\nu_0, \ldots, \nu_0]$, so by definition $(\mu', \nu') \in \mathcal{B}_0 \subseteq \mathcal{B}$. Using again transitivity, we conclude that $(\mu', \nu) \in \mathcal{B}^*$, which is enough for our purposes, since $\nu \rightarrow^* \nu$. This settles the matter as far point (2) is concerned; for point (4), we have



We already know that $\mu \rightarrow \mu'$ as above; it is not hard to see that $\mu' \rightarrow^* C_0[\mu_0, \ldots, \mu_0]$. Now $(C_0[\mu_0, \ldots, \mu_0], C_0[\nu_0, \ldots, \nu_0])$ is in $\mathcal{B}_0$ by definition, so we are done.

We get to the case $\alpha = \delta_2$. The reduction $\mu \rightarrow \mu'$ looks as follows:

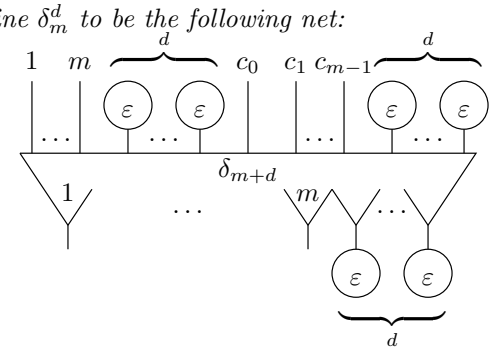


Notice that, if we take all of μ' except the copies of μ_0 which have not interacted, we can form a multi-hole context and write $\mu' = C'[\mu_0, \ldots, \mu_0]$. Now we take ν, and observe that, with the help of Lemma 6.1, we have just as above $\nu \simeq_\eta \dot{\approx} C'[\nu_0, \ldots, n_0]$; then we conclude by inclusion of η-equivalence and barbed bisimulation in $\mathcal{B}$, and by transitivity. The case in which $\nu \rightarrow \nu'$ is also handled just as above. This takes care of the situation in which the cell δ_2 interacts with μ_0 or ν_0 through its first principal port; the case of the second principal port is perfectly analogous, so there is nothing else to check as far as the above equation is concerned.

There are two more η-equations to verify, each verification consisting of the analysis of several subcases. However, there is nothing essentially new with respect to what we have shown so far; the techniques employed above are used over and over, with our definition of $\mathcal{B}$ (including $\simeq_\eta$ and $\dot{\approx}$) and transitivity of $\mathcal{B}^*$ playing fundamental rôles. Therefore, we shall stop here, inviting the curious (and well motivated) reader to check a few of the remaining cases. □

We immediately show an application of Lemma 6.2, which will be useful in the encoding of generic 2INS's into the 2-port combinators:

Lemma 6.3 *Define δ_m^d to be the following net:*



Then, $\delta_1^1 \cong \delta_1$.

Proof. Simply observe that δ_1^1 reduces to δ_1 by means of deterministic steps only, so the result follows from Lemma 6.2. □

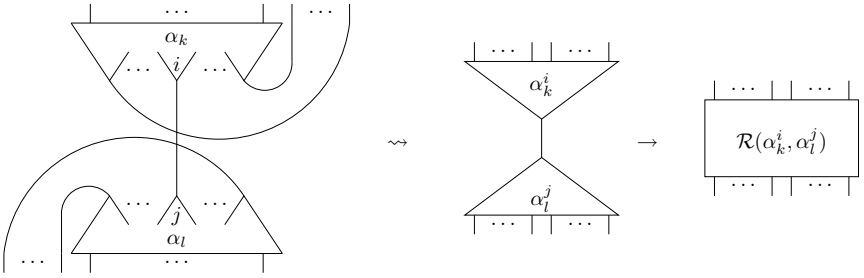
6.4 Universality

In the following, we fix a generic multiport interaction net system $\mathcal{S} = (\Sigma, \mathcal{R})$ such that $\Sigma = \{\alpha_1, \ldots, \alpha_c\}$. We let n_i and m_i denote resp. the arity and coarity of α_i, and we pose $m = \max\{m_1, \ldots, m_c\}$.

We shall define a function $[\cdot]$ mapping nets of $\mathcal{S}$ to nets of m-port combinators, for all $m \geq 1$. Since we have defined a behavioral equivalence only for the 2-port combinators, we shall be able to prove that $[\cdot]$ is a translation (up to $\cong$) only in the 2-port case. This is already quite satisfying in the light of the results of Sect. 5.4: we know that there exists a 2INS capable of faithfully encoding the "core" π-calculus, so the 2-port combinators have at least the same expressive power as this subcalculus of π, which is not bad. By the way, we strongly believe our translation to be sound for any number of principal ports; what is missing is just a good notion of behavioral equivalence for generic multiport combinators, which would allow us to prove it.

6.4.1 Projections and decomposition of rules

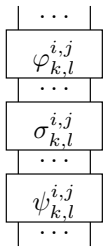
To each cell α_k we can associate m_k unicells $\alpha_k^1, \ldots, \alpha_k^{m_k}$, each of arity $n_k + m_k - 1$, representing the different behaviors of α_k according to the principal port chosen for interaction. The α_k^i are called the *projections* of α_k. Multiport interaction rules can be decomposed in terms of projections; we first select the suitable projections, then we apply the rule:



Of course the rewriting step denoted by $\rightsquigarrow$ is not an interaction rule (the right hand member is not cut-free); the interest of the above decomposition is that we can see multiport reduction as a non-deterministic choice plus a normal interaction net reduction.

The right member of interaction rules can be canonically decomposed, as done by Lafont [Laf97]:

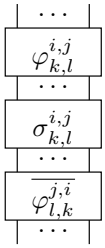
$\mathcal{R}(\alpha_k^i, \alpha_l^j) \;=$



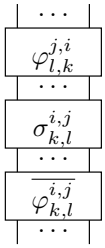
where $\varphi_{k,l}^{i,j}$ and $\psi_{k,l}^{i,j}$ do not contain active pairs, and $\sigma_{k,l}^{i,j}$ is a permutation. From this, and by the symmetry condition on right members of rules, we obtain that $\varphi_{l,k}^{j,i} = \overline{\psi_{k,l}^{i,j}}$, and $\sigma_{l,k}^{j,i} = \overline{\sigma_{k,l}^{i,j}}$. In particular, $\overline{\sigma_{k,k}^{i,i}} = \sigma_{k,k}^{i,i}$. Now, in case $i \neq j$, we can always include the permutation $\sigma_{k,l}^{i,j}$ into $\varphi_{k,l}^{i,j}$, so that the decomposition is no longer canonical, but we have that the permutation "between" the two components of the decomposition of $\mathcal{R}(\alpha_k^i, \alpha_l^j)$ is the identity.

Therefore, we can in general associate to the rule involving α_k^i and α_l^j two cut-free nets $\varphi_{k,l}^{i,j}, \varphi_{l,k}^{j,i}$ and one permutation $\sigma_{k,l}^{i,j}$ satisfying $\overline{\sigma_{k,l}^{i,j}} = \sigma_{k,l}^{i,j}$, such that

$\mathcal{R}(\alpha_k^i, \alpha_l^j) \;=$



$\mathcal{R}(\alpha_l^j, \alpha_k^i) \;=$



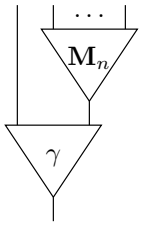
6.4.2 Multiplexors

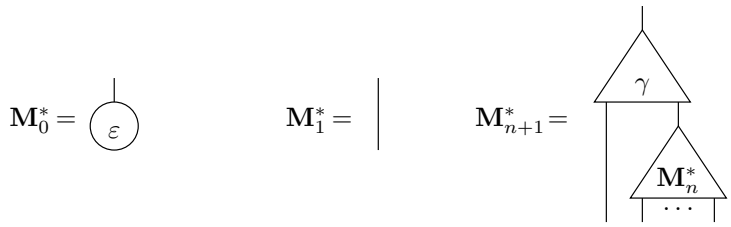
For each $n \geq 0$, we define the principal nets (they are actually trees) $\mathbf{M}_n$ and $\mathbf{M}_n^*$ as follows:

$\mathbf{M}_0 =$

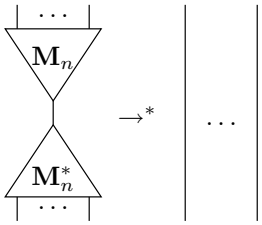
$\mathbf{M}_1 =$

$\mathbf{M}_{n+1} =$

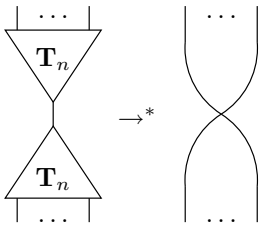




We invite the reader to check that following property holds, for all $n \geq 0$:



Using δ_1 cells instead of γ cells for any one of the above constructions, for example the one defining the $\mathbf{M}_n$ family, yields a family of trees $\mathbf{T}_n$ for all $n \geq 0$, having the following property:

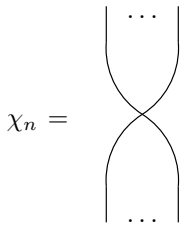


6.4.3 Rotation-invariant wirings

We define, for all $n \geq 1$, the permutation on n elements χ_n such that

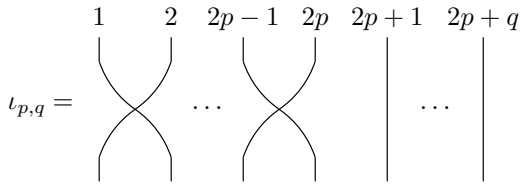
$$\chi_n(i) = n - i + 1, \ i \in \{1, \ldots, n\}.$$

As a wiring, χ_n is a made of n wires arranged as follows:



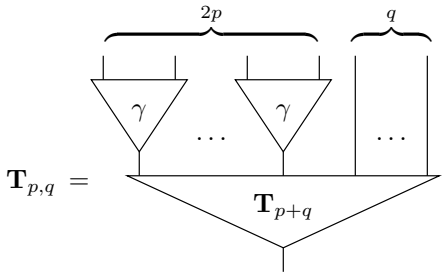
Clearly χ_n is involutive, i.e., $\chi_n^2 = I$, where I is the identity permutation.

We also define, for all non-negative p, q such that $2p+q \geq 1$, the permutation $\iota_{p,q}$ on $2p + q$ elements exchanging 1 with 2, 3 with 4, and so on until $2p - 1$ with $2p$, and behaving as the identity on $2p+1, \ldots, 2p+q$. In terms of wirings, we have

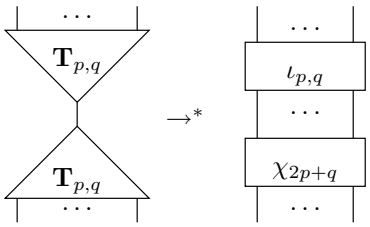


Notice that any involutive permutation σ on n elements is a composition of disjoint transpositions, therefore $\sigma = \rho^{-1}\iota_{p,q}\rho$ for some permutation ρ and some p, q such that $2p + q = n$.

For all non-negative p, q such that $2p + q \geq 1$, we define a tree $\mathbf{T}_{p,q}$ (called *transpositor*) of arity $2p + q$ as follows:



The reader can check that transpositors enjoy the following property:



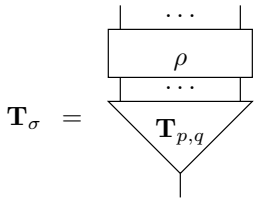
This is possible because, if we pose $n = 2p + q$, $\chi_n\iota_{p,q}$ is rotation-invariant, i.e., $\overline{\chi_n\iota_{p,q}} = \chi_n\iota_{p,q}$ (we recall that, for any net μ, $\overline{\mu}$ is the same net rotated 180 degrees, cf. Sect. 1.1.2). In fact, given a permutation σ on n elements, the permutation represented by the wiring $\overline{\sigma}$ can be defined as $\overline{\sigma} = \chi_n\sigma^{-1}\chi_n$. Since χ_n and $\iota_{p,q}$ are both involutive, we have $(\chi_n\iota_{p,q})^{-1} = \iota_{p,q}\chi_n$, so that

$$\overline{\chi_n\iota_{p,q}} = \chi_n(\chi_n\iota_{p,q})^{-1}\chi_n = \chi_n\iota_{p,q}\chi_n^2 = \chi_n\iota_{p,q}.$$

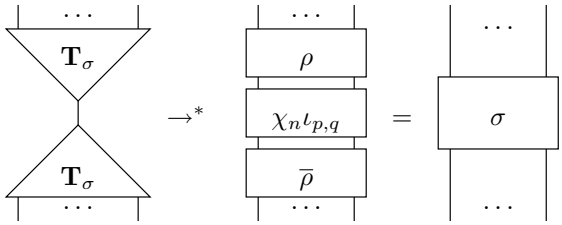
Now let σ be a rotation-invariant wiring/permutation on n elements. By the above discussion, and by the involutivity of χ_n, we have $\chi_n\sigma = \sigma^{-1}\chi_n$, which means that $\chi_n\sigma$ is involutive. But then there exist p, q such that $2p + q = n$ and a permutation on n elements ρ such that $\chi_n\sigma = \rho^{-1}\iota_{p,q}\rho$. Using again the involutivity of χ_n, we have

$$\sigma = \chi_n\rho^{-1}\iota_{p,q}\rho = (\chi_n\rho^{-1}\chi_n)(\chi_n\iota_{p,q})\rho = \overline{\rho}(\chi_n\iota_{p,q})\rho.$$

Therefore, if we pose

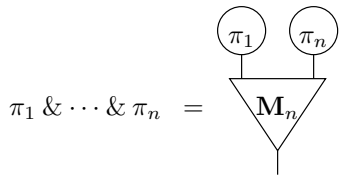


by the property of transpositors we obtain a principal net capable of generating σ:

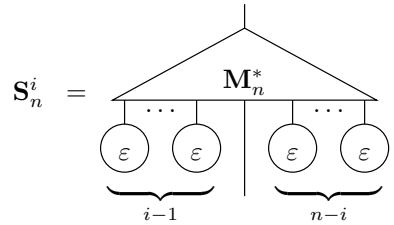


6.4.4 Menus and selectors

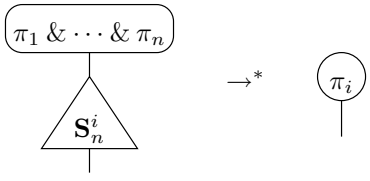
If we have n packages $\pi_1, \ldots, \pi_n$, we can build another package $\pi_1 \,\&\, \cdots \,\&\, \pi_n$, called *menu*, defined as follows:



Then, for all $n \geq 1$ and $1 \leq i \leq n$, we define the *selector*



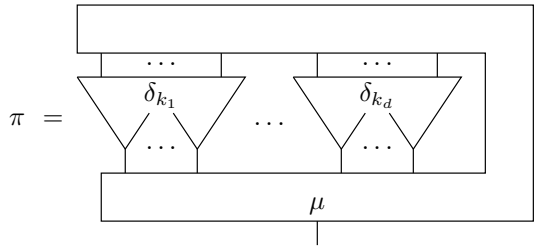
We invite the reader to check that the following holds:



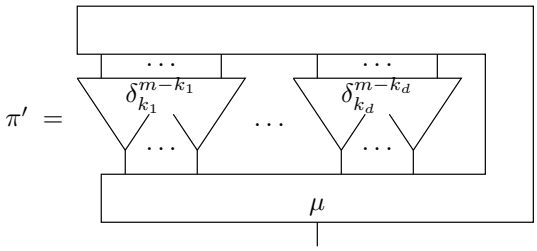
6.4.5 Codes, decoders, and copiers

The following is the fundamental construction for encoding arbitrary mINS's into the multiport combinators, and is an adaptation of Lafont's "bang" construction [Laf97].

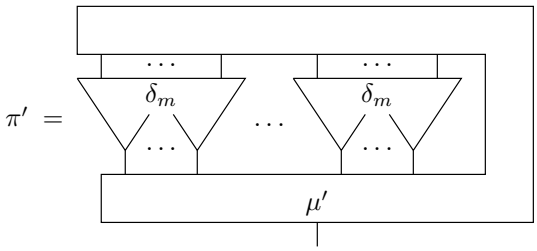
Consider a generic package π built with m-port combinators. Suppose that π contains d δ_k cells (with $1 \leq k \leq m$); then we can write



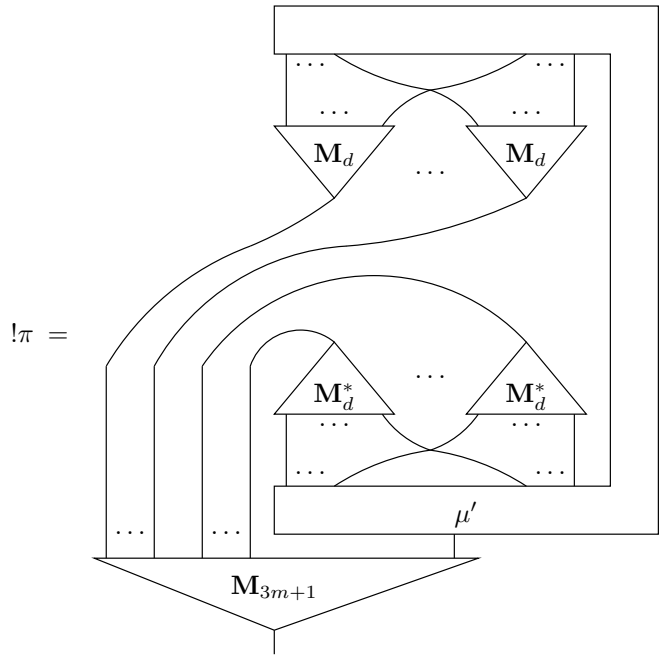
Using the nets defined in Lemma 6.3, we modify π as follows:



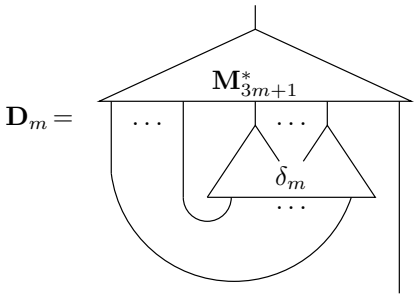
Now all δ_k cells contained in π' are such that $k = m$, i.e., we have



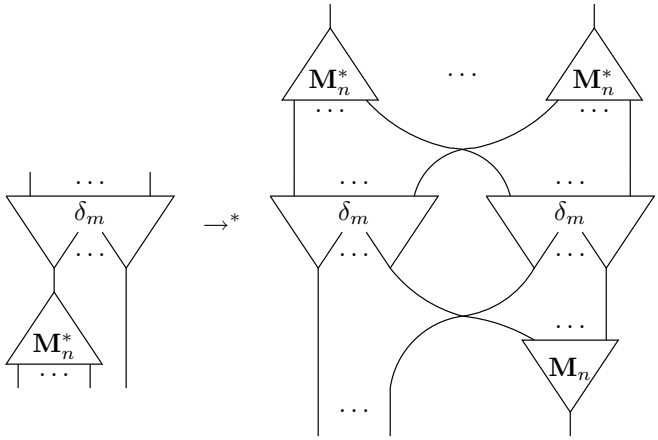
Then, we define the *code* of π to be the following package:



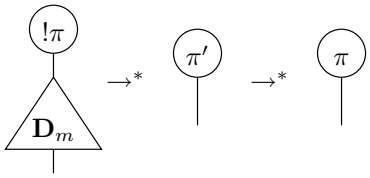
Now we define the *decoder* net as follows:



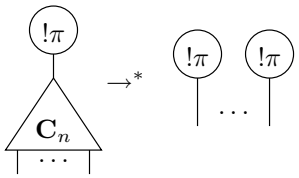
It can be proved (by induction) that the following holds, for all m, n:



Using this, it is not hard to check that



Packages made up of γ and ε cells can be duplicated by δ_1 cells (this is easily proved by induction). If we put $\mathbf{C}_n = \mathbf{T}_n$ for all $n \geq 0$, we have

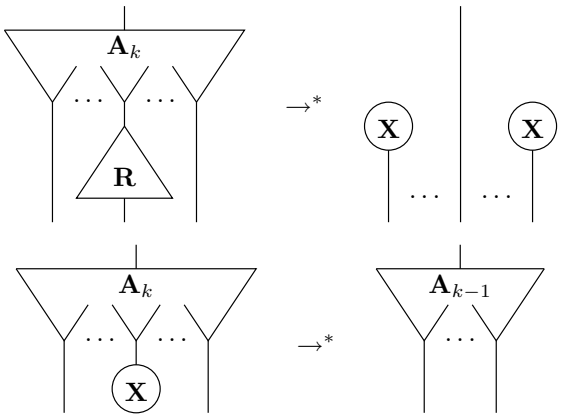


The nets $\mathbf{C}_n$ will be called *copiers*.

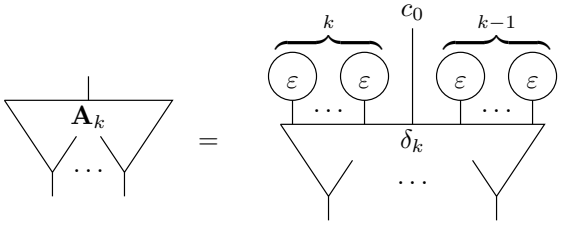
6.4.6 Arbiters and claimers

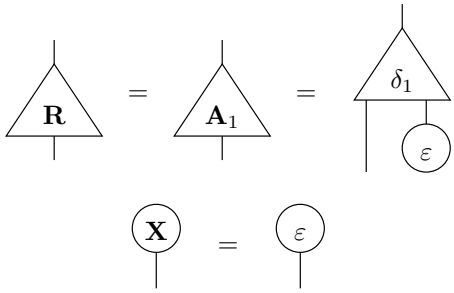
The one we introduced here is the only construction which is not already present (at least in a deterministic form) in Lafont's work. We want to build, for all $k \geq 1$, an *arbiter* net $\mathbf{A}_k$, which listens on k channels for concurrent requests coming from *claimers* $\mathbf{R}$. The arbiter non-deterministically chooses a claimer, and assigns to it the resource it is handling; the other claimers receive a stop signal $\mathbf{X}$. Stop signals may also be sent to the arbiter, in order to disable one of its channels.

The above description corresponds to the following reductions:



We invite the reader to check that a possible (particularly economic) implementation of the above nets is the following:

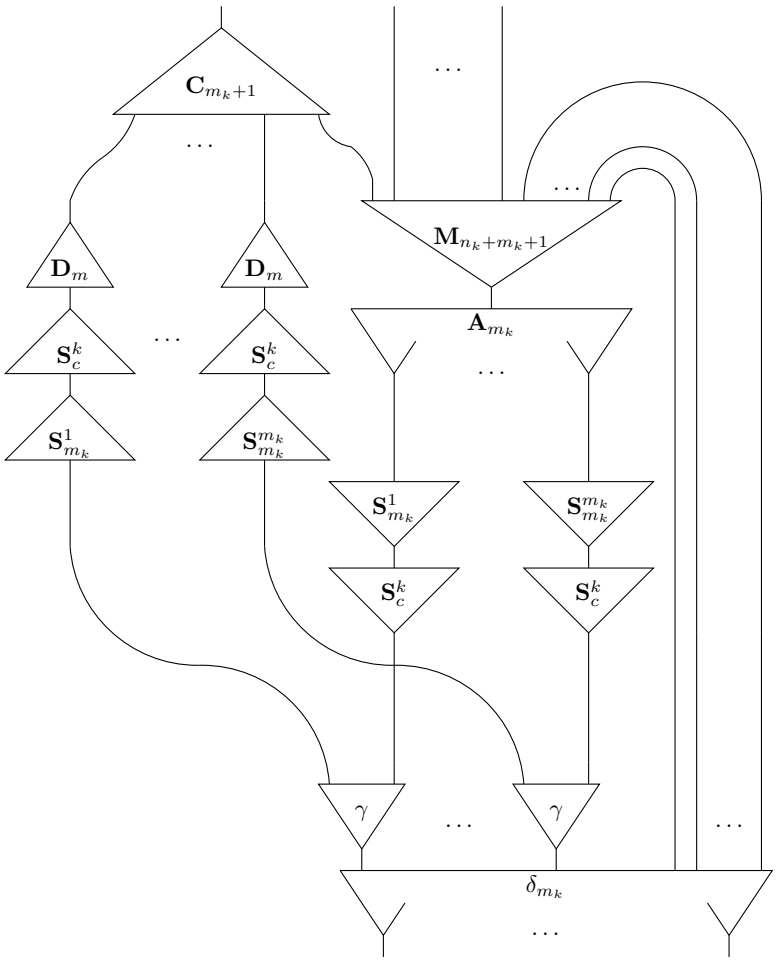




6.4.7 The translation

We now define our translation from $\mathcal{S}$ to the m-port combinators. First of all, set

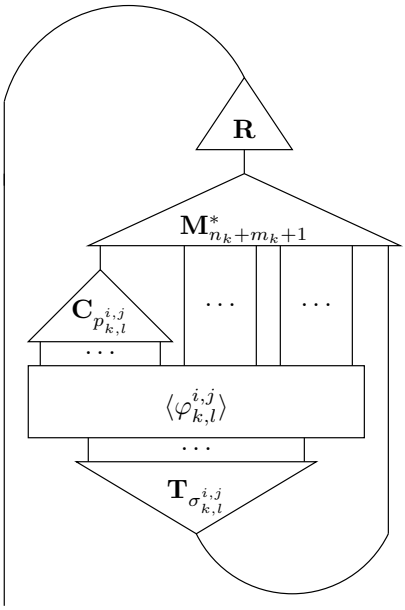
$\langle \alpha_k \rangle$ =



The function $\langle \cdot \rangle$ cannot be a translation because it is not arity-preserving: cells are mapped to nets having an extra free port. Nevertheless, $\langle \cdot \rangle$ can be extended without problems to all nets of $\mathcal{S}$; given a net μ with n free ports containing p cells, $\langle \mu \rangle$ has $n + p$ free ports.

So let $p_{k,l}^{i,j}$ be the number of cells contained in the net $\varphi_{k,l}^{i,j}$. For all $k, l \in \{1, \ldots, c\}$ and for all $i \in \{1, \ldots, m_k\}$, $j \in \{1, \ldots, m_l\}$, we define the package $\Gamma_{k,l}^{i,j}$ as follows:

$\Gamma_{k,l}^{i,j} \quad =$

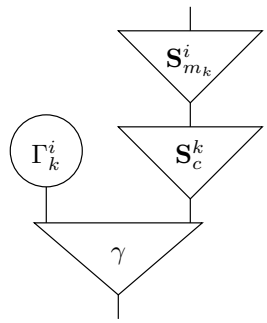


where the copier $\mathbf{C}_{p_{k,l}^{i,j}}$ collects all the extra free ports of $\langle \varphi_{k,l}^{i,j} \rangle$ introduced by the application of $\langle \cdot \rangle$. Using the $\Gamma_{k,l}^{i,j}$ packages just defined, we build the following packages:

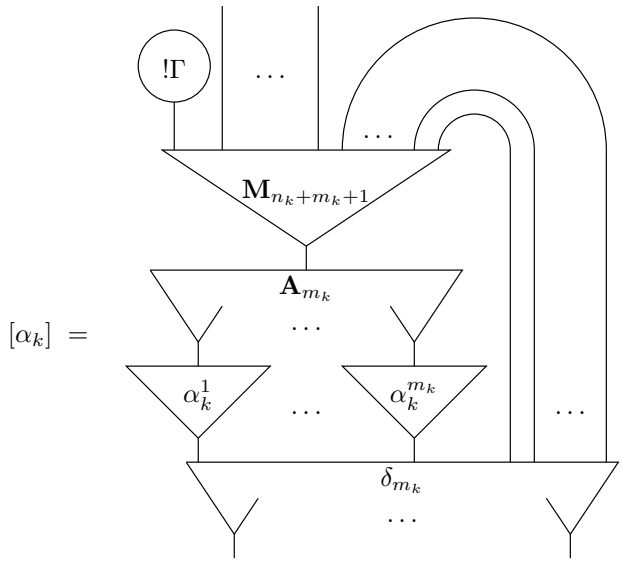
$$\begin{aligned} \Gamma_{k,l}^{i} &= \Gamma_{k,l}^{1,i} \,\&\, \cdots \,\&\, \Gamma_{k,l}^{m_k,i} \\ \Gamma_{k}^{i} &= \Gamma_{1,k}^{i} \,\&\, \cdots \,\&\, \Gamma_{c,k}^{i} \\ \Gamma_{k} &= \Gamma_{k}^{1} \,\&\, \cdots \,\&\, \Gamma_{k}^{m_k} \\ \Gamma &= \Gamma_{1} \,\&\, \cdots \,\&\, \Gamma_{c} \end{aligned}$$

For all $k \in \{1, \cdots, c\}$ and $i \in \{1, \ldots, m_k\}$, we pose

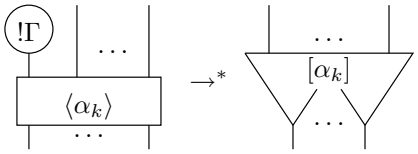
$\alpha_k^i =$



We are now ready to define the translation $[\cdot]$ from $\mathcal{S}$ to the m-port combinators:



The function $\langle\cdot\rangle$ is related to the translation by the following property, which we invite the reader to verify:

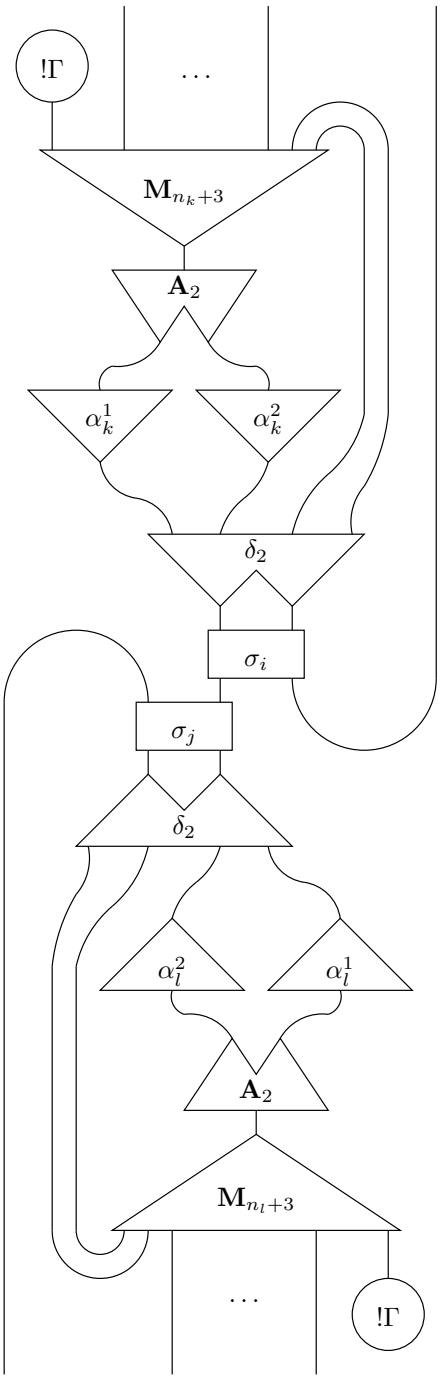


We remark that, if $\mathcal{S}$ is an INS, then the translation has the interaction combinators as target system; indeed, in that case if one replaces $\mathbf{A}_1$ and $\mathbf{R}$ cells by simple wires (a logical optimization, since they are not needed anymore), $[\cdot]$ becomes exactly Lafont's translation.

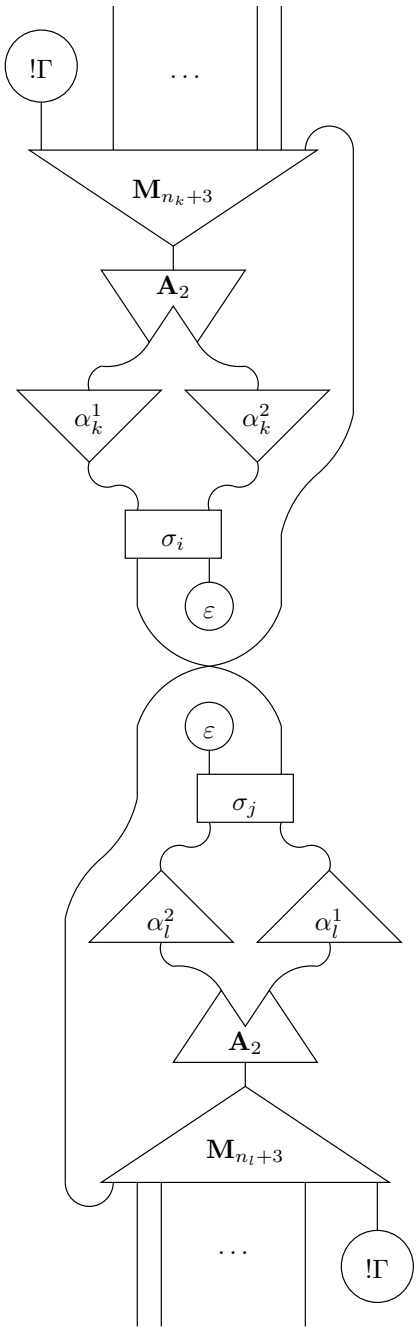
Using the above constructions, one can prove the following:

Theorem 6.4 (Universality) *Any 2INS can be translated up to* $\cong$ *into the* 2*-port combinators.*

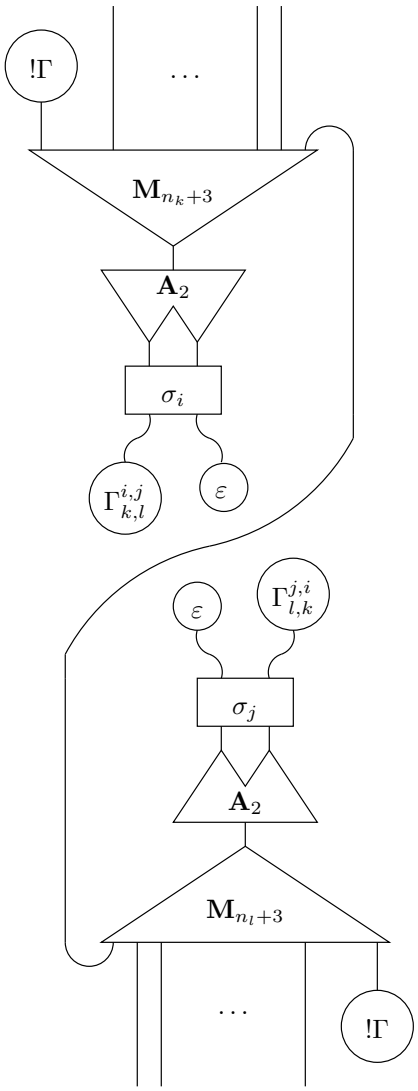
Proof. Take a generic active pair $\alpha_k^i \bowtie \alpha_l^j$ of $\mathcal{S}$. Its translation, which we call μ, is



where σ_i is the identity if $i = 1$, or the "twist" if $i = 2$, and similarly for σ_j. After one interaction step, μ reduces to a net which we call μ', and which is equal to

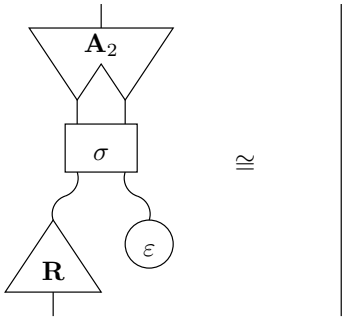


Now the selectors in $[\alpha_k]$ extract first $\Gamma^j_{k,l}$ from Γ^j_l and then $\Gamma^{i,j}_{k,l}$ from this latter, while the selectors in $[\alpha_l]$ progressively extract $\Gamma^{j,i}_{l,k}$ from Γ^i_k. Meanwhile, the ε combinators delete the α^p_k and α^q_l which have not been chosen by the interaction. Notice that selectors use only γ and ε interaction to extract the desired packages, while ε combinators erase cells by means of deterministic reductions only; therefore, by Lemma 6.2, the above net is barbed congruent to



The $\Gamma^{i,j}_{k,l}$ and $\Gamma^{j,i}_{l,k}$ packages "start" with an **R** net, and hence with a δ_1 cell. This means that at this moment we may choose to perform a non-deterministic reduction. But this is not harmful, as there is a deterministic reduction sequence having the same result:

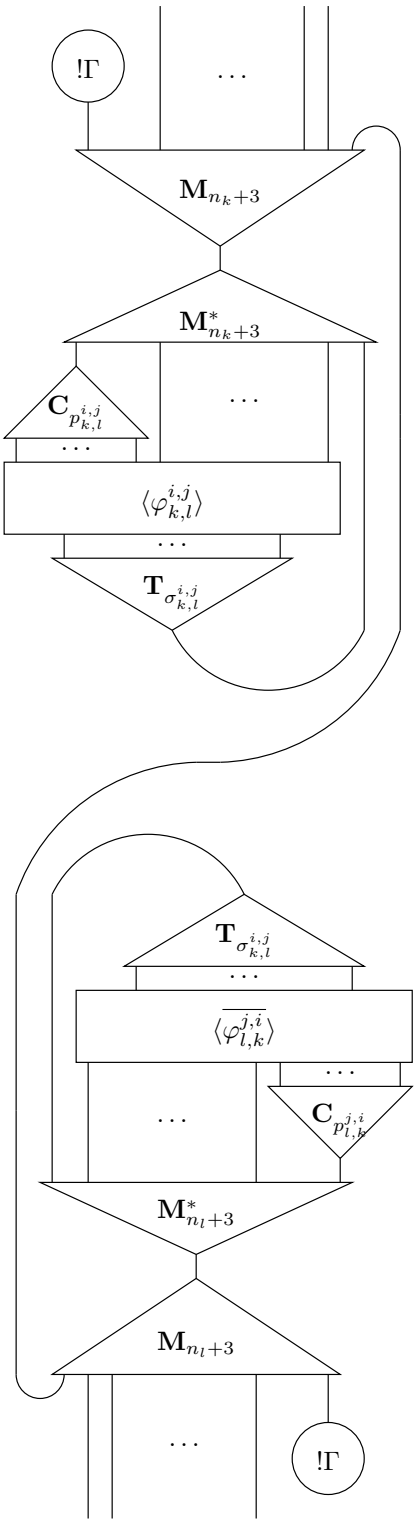
Claim 6.4.1 *The following holds:*



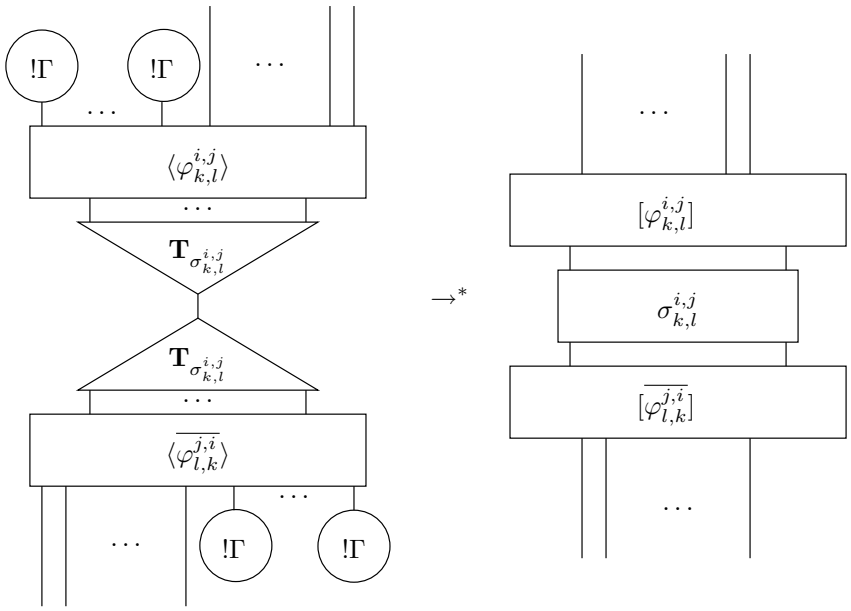
where σ is either the identity or the "twist" permutation.

Proof. Simply consider that the wire can be obtained by first making the ε cell interact with the δ_2 cell in $\mathbf{A}_2$, which yields a δ_1 that can at this point interact with the δ_1 cell in the $\mathbf{R}$ net; the wire is then obtained after a few $\varepsilon\varepsilon$ interactions. This reduction sequence is composed of deterministic steps only, hence the claim (by Lemma 6.2). Of course making the δ_1 cell in $\mathbf{R}$ directly interact with the δ_2 cell in $\mathbf{A}_2$ also yields a wire (after a few $\varepsilon\varepsilon$ steps); we have simply shown that this non-deterministic interaction does not actually make any choice, as the net is in some sense "destined" to converge to a wire. □

Using Claim 6.4.1, we know that μ' is barbed congruent to the following net:



which, by the properties of the constructions shown above, reduces to



which is equal to $[\mathcal{R}(\alpha^i_k, \alpha^j_l)]$. The reader can check that copying a $!\pi$ package and decoding it is done by means of deterministic reductions only, so we obtain $[\alpha^i_k \bowtie \alpha^j_l] = \mu \to \mu' \cong [\mathcal{R}(\alpha^i_k, \alpha^j_l)]$ by applying Lemma 6.2 and the transitivity of $\cong$. □

Bibliography

[Ale99] Vladimir Alexiev. *Non-deterministic Interaction Nets*. Ph.D. Thesis, University of Alberta, 1999.

[Bec98] Denis Bechet. **Minimality of the correctness criterion for multiplicative proof nets**. *Mathematical Structures in Computer Science*, 8(6):543–558, December 1998.

[BM05] Emmanuel Beffara and François Maurel. **Concurrent Nets: a study of prefixing in process calculi**. In *Proceedings of EXPRESS 2004*, volume 128 of *ENTCS*, pages 67–86. Elsevier, April 2005.

[DR89] Vincent Danos and Laurent Regnier. **The Structure of Multiplicatives**. *Archive for Mathematical Logic*, 28:181–203, 1989.

[DR95] Vincent Danos and Laurent Regnier. **Proof nets and the Hilbert space**. In J.-Y. Girard, Y. Lafont, and L. Regnier, editors, *Advances in Linear Logic*, pages 307–328. Cambridge University Press, 1995.

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

[EL06] Thomas Ehrhard and Olivier Laurent. **On Differential Interaction Nets and the Pi-calculus**. Technical report, Preuves, Programmes et Systèmes, 2006.

[ER06] Thomas Ehrhard and Laurent Regnier. **Differential Interaction Nets**. to appear in *Theoretical Computer Science*, 2006.

[FM03] Maribel Fernández and Ian Mackie. **Operational Equivalence for Interaction Nets**. *Theoretical Computer Science*, 297(1–3):157–181, 2003.

[Fu98] Yuxi Fu. **Reaction Graph**. *Journal of Computer Science and Technology*, 13(6):510–530, 1998.

[Gir87a] Jean-Yves Girard. **Linear Logic**. *Theoretical Computer Science*, 50(1):1–102, 1987.

[Gir87b] Jean-Yves Girard. **Multiplicatives**. In G. Lolli, editor, *Logic and Computer Science: New Trends and Applications*, pages 11–34, 1987. Rendiconti del Seminario Matematico dell'Università e Politecnico di Torino.

[Gir89] Jean-Yves Girard. **Geometry of Interaction I: interpretation of System F**. In *Proceedings of the Logic Colloquium '88*, pages 221–260. North Holland, 1989.

[Gir01] Jean-Yves Girard. **Locus Solum**. *Mathematical Structures in Computer Science*, 11(3):301–506, 2001.

[Hug05] Dominic Hughes. **Simple multiplicative proof nets with units**. June 30 2005. Submitted for publication.

[Hyl76] Martin Hyland. **A Syntactic Characterization of the Equality in Some Models of the Lambda Calculus**. *J. London Math. Society*, 2(12):361–370, 1976.

[Kha03] Lionel Khalil. *Généralisation des Réseaux d'Interaction avec* amb*, l'agent de McCarthy: propriétés et applications*. Ph.D. Thesis, École Normale Supérieure de Paris, 2003.

[Kri90] Jean-Louis Krivine. *Lambda-calcul, types et modèles*. Masson, Paris, 1990.

[Laf90] Yves Lafont. **Interaction Nets**. In *Conference Record of POPL'90*, pages 95–108. ACM SIGACT and SIGPLAN, ACM Press, 1990.

[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.

[Laf97] Yves Lafont. **Interaction Combinators**. *Information and Computation*, 137(1):69–101, 1997.

[Laf98] Yves Lafont. **Introduction to interaction nets**. Manuscript, 1998. Available at `http://iml.univ-mrs.fr/~lafont/pub/nets.ps.gz`.

[Lip02a] Sylvain Lippi. **Encoding Left Reduction in the Lambda-Calculus with Interaction Nets**. *Mathematical Structures in Computer Science*, 12(6):797–822, 2002.

[Lip02b] Sylvain Lippi. *Théorie et pratique des réseaux d'interaction*. Ph.D. Thesis, Université de la Méditerranée, 2002.

[LPV01] Cosimo Laneve, Joachim Parrow, and Björn Victor. **Solo Diagrams**. In *Proceedings of TACS'01*, volume 2215 of *Lecture Notes in Computer Science*, pages 127–144. Springer-Verlag, 2001.

[LV99] Cosimo Laneve and Björn Victor. **Solos in Concert**. In *Proceedings of ICALP'99*, volume 1644 of *LNCS*, pages 513–523. Springer-Verlag, 1999.

[Mil94] Robin Milner. **Pi-Nets: A Graphical Form of π-Calculus**. In *Proceedings of ESOP'94*, volume 788 of *Lecture Notes in Computer Science*, pages 26–42. Springer, 1994.

[Mor98] Andrew Moran. *Call-by-name, Call-by-need, and McCarthy's Amb*. Ph.D. Thesis, University of Gasgow, 1998.

[MP02] Ian Mackie and Jorge Sousa Pinto. **Encoding Linear Logic with Interaction Combinators**. *Information and Computation*, 176(2):153–186, 2002.

[Pag06] Michele Pagani. **Proofs, denotational semantics, and observational equivalence in multiplicative linear logic**. To appear in *Mathematical Structures in Computer Science*, 2006.

[Par95] Joachim Parrow. **Interaction Diagrams**. *Nordic Journal of Computing*, 2:407–443, 1995. A previous version appeared in *Proceedings of A Decade in Concurrency*, LNCS 803: 477–508, 1993.

[PS88] Prakash Panangaden and Vasant Shanbhogue. **McCarthy's Amb Cannot Implement Fair Merge**. In *Foundations of Software Technology and Theoretical Computer Science*, pages 348–363, Berlin - Heidelberg - New York, December 1988. Springer.

[PT97] Benjamin Pierce and David Turner. **Pict: A Programming Language Based on the Pi-Calculus**. CSCI Technical Report 476, Indiana University, March 1997.

[SW01] Davide Sangiorgi and David Walker. *The π-calculus — A Theory of Mobile Processes*. Cambridge University Press, 2001.

[Wad76] Christopher Wadsworth. **The Relation Between Computational and Denotational Properties for Scott's D_∞ Models**. *Siam J. Comput.*, 5(3):488–521, 1976.

[Yos95] Nobuko Yoshida. **Graph Notation for Concurrent Combinators**. In *Proceedings of TPPP'99*, volume 907 of *LNCS*, pages 393–412. Springer, 1995.