The Geometry of Concurrent Interaction: Handling Multiple Ports by Way of Multiple Tokens

Ugo Dal Lago, Ryo Tanaka, Akira Yoshimizu

The Geometry of Concurrent Interaction: Handling Multiple Ports by Way of Multiple Tokens (Long Version)

Ugo Dal Lago Ryo Tanaka Akira Yoshimizu

Abstract

We introduce a geometry of interaction model for Mazza's multiport interaction combinators, a graph-theoretic formalism which is able to faithfully capture concurrent computation as embodied by process algebras like the π-calculus. The introduced model is based on token machines in which not one but *multiple* tokens are allowed to traverse the underlying net *at the same time*. We prove soundness and adequacy of the introduced model. The former is proved as a simulation result between the token machines one obtains along any reduction sequence. The latter is obtained by a fine analysis of convergence, both in nets and in token machines.

1 Introduction

Game semantics [3,26] and the geometry of interaction (GoI for short) [13,23] are semantic frameworks in which programs and proofs are interpreted as mathematical or computational objects exhibiting nontrivial interactive behaviours (e.g. strategies [26], token machines [13], operators [23]). This allows for a number of nice properties. First of all, these models can be defined so as to be *compositional*, but close to *contexts* as for their discriminating power; this is particularly true in game semantics, for which many full abstraction results have been proved. Secondly, interactive models can often be presented concretely, either as circuits [20, 22], automata [13], or abstract machines for strategies [19], thus enabling direct compilation of higher-order programs into low-level languages.

The bulk of the huge amount of literature on interactive semantic models is about sequential languages [3, 26] but, especially in the last fifteen years, the research community has been able to devise game models sufficiently powerful to interpret not only advanced features like polymorphisms and general references [1, 2], but also concurrency [4, 8, 34, 37], thus going significantly beyond game semantics as originally conceived (more on this is in Section 1.1 below). The same cannot be said about the geometry of interaction, which until very recently has been able to interpret only sequential, although potentially effectful, forms of computation [25, 35].

A crucial observation, which is the starting point of this work, is that the geometry of interaction, when formulated in terms of so-called token machines, can be made parallel by allowing *more than* a single token to float around *at the same time* [9]. The consequences of this idea have been analysed for sequential languages [10], also in presence of probabilistic and quantum effects [11]. Are multiple tokens enough to model fully-fledged concurrency, as embodied by process algebras? This is the question we will try to address in this paper. The answer will be positive, although the walk to it will not be easy.

When looking for a GoI semantics for concurrent models of computation, one could of course proceed by considering any concrete process algebra, and define a token machine for it *directly*, being inspired by the literature. As an example, the way the higher-order π-calculus is classically encoded into the usual name-passing π-calculus [38] can be seen reminiscent of the usual GoI construction, since higher-order process passing is encoded into a somehow more basic, essentially first-order calculus. This route would however be biased to a specific process algebra, thus losing in generality and canonicity. In turn, relying on a (possibly complicated) computational model would mean hiding the *structure* of the introduced GoI, understanding what is our main aim here.

For these reasons, we will purposely take a minimalistic approach, being inspired by Lafont's *interaction nets* (*INs* for short) and *interaction combinators* (for short, *ICs*) [27]. The latter is a system of interaction nets which is *universal*, and thus embodies a vast class of graph-theoretic models for sequential interaction, including logical systems [24], programming languages [30], and optimal reduction algorithms [29]. ICs are not only very simple themselves, but also admit an elementary geometry of interaction model, arguably the simplest of all [27]. ICs, however, cannot form the basis on which to build a GoI model of concurrent computation: they are strongly confluent and simply lack the mixture of parallelism and nondeterminism which is at the heart of concurrency, although being a very good model of (low-level) sequential computation.

A picture similar to the one drawn by Lafont but envisioned with concurrency in mind is the one due to Alexiev, Mazza, and coauthors [5, 15, 32, 33]. In the last twenty years, in particular, concurrent extensions of interaction nets, called *multiport* interaction nets (*MINs* for short), have been proved to be powerful enough to faithfully encode process algebras [32, 33], to admit an event-structure model [31], and to be strictly more expressive than multi*rule* interaction nets [15]. Remarkably, MINs have also been shown to admit a universality theorem with respect to a specific interaction system, namely *multiport interaction combinators* (for short, *MICs*) [33]. In a sense, then, we have the following equation:

$$\frac{INs}{ICs} = \frac{MINs}{MICs}.$$

MINs generalise INs in that cells can have *more than one* principal port, this way allowing for a very general form of nondeterministic interaction, and MICs, being universal for MINs, are a natural candidate model for studying GoI models for concurrent systems, as suggested by Mazza himself [33]. Nevertheless, we are not aware of any attempt to give geometry of interaction models for any system of MINs.

When trying to define a GoI model for MICs, one immediately realises that classic token machines are simply *not* adequate to faithfully model concurrent interaction. In particular, multiport interaction, i.e., nondeterminism as found in MICs, cannot be captured by automata in which just one token is allowed to float around the underlying graph, as will be explained in Section 2.3 below. The only way out consists in allowing for the simultaneous presence of multiple tokens [9, 10], which becomes essential here. In particular, it allows for *nondeterministic* and *nonlocal* interaction, which is an essential ingredient of MICs' dynamics, and of concurrency in general.

This paper is devoted to presenting the first geometry of interaction model for multiport interaction combinators. Our GoI model is a substantial extension of Lafont's classic token machine model for ICs [27]. We allow multiple tokens to move around at the same time and make use of them to realise the process of resolving nondeterminism and keeping track of choices, that we call *marriages*. This requires four different kinds of token, three of them being static, and only one meant to really travel inside the net. This way, we get a stateless notion of a machine even if, of course, static tokens could be replaced by stateful cells.

Our *Multi-token machines* can be seen as *locative transition systems*, a special class of labelled transition systems. This framework provides us with a natural way of defining parallel composition, and we will prove our semantics to be indeed compositional, i.e., that the parallel composition of two nets can be interpreted as the parallel composition of the two respective machines, modulo bisimilarity. We can thus establish soundness of the model in terms of labelled (bi)similarity. What makes everything much more complex than in single-token machines is the nondeterministic nature of the reduction system: a net can make nondeterministic choices, thus losing the capability of behaving in a certain way. If a net ρ reduces to another net π by performing such a reduction step, then the interpretation of ρ is not necessarily behaviourally *equivalent* to that of π, but can rather be "larger". As a consequence, soundness of our GoI model needs to be spelled out in the form of *similarity* which, by the way, turns out to come from bisimilarity between other associated states of the two involved machines.

We also show that our model is not too coarse but *adequate*, in the sense that token machines reflect the convergence behaviour of the nets they interpret, both in the "may" and in the "must" sense. Our proof heavily exploits the bisimulation relations we use to prove our soundness result, which can relate an execution of a token machine to another one along net reduction.

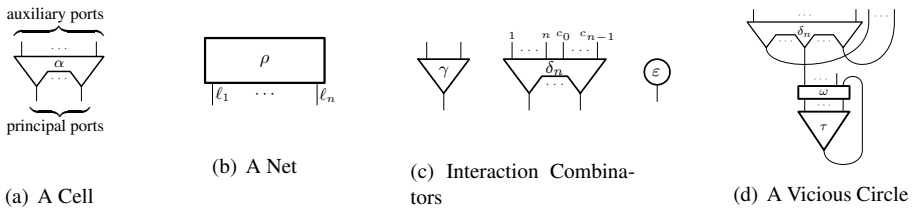


Figure 1: Cells, Nets, Combinators, and Vicious Circles.

1.1 Related Work

Concurrent (or asynchronous) game semantics, first introduced in [4] and pursued later by Melliés [34], is a generalisation of usual game semantics for sequential computation [3, 26]. It yielded a fully abstract model of multiplicative additive linear logic proofs, followed by (again fully abstract) models of many concurrent calculi, e.g. CSP or Parallel Algol [21, 28]. Recently, Winskel and his coauthors have studied winning conditions and determinacy results for such games, that may lead to applications in verification of concurrent systems [8, 37]. We are not aware of any attempt to relate all this to geometry of interaction and token machines.

Another formalism of concurrent and distributed systems that is worth mentioning here is the one of Petri nets [36]. Indeed, our token machines resemble Petri nets to a large extent: multiple tokens circulate around a graph structure, dynamically enabling or disabling each other's transition. However, there is one remarkable difference: while in Petri nets the underlying graph consists itself of places and transitions and computation is inherently *local*, out token machines indeed allow tokens lying next to cells which are far away from each other in a net to communicate, meaning that interaction is *non*local. This is an inevitable price to pay if we want token machines to properly reflect the behaviour of the MIC reduction rules. See Section 2.3 for the details, and Section 7 for more observations in this direction.

Differential interaction nets [18] are a graphical calculus for differential linear logic [16], and can be seen as a multi*rule* variant of interaction nets, for which a single token GoI model already exists [14]. They exhibit nondeterministic behaviour and are able to encode finitary fragments of Milner's π-calculus [17], although the encoding *cannot* be completely satisfactory, as highlighted by Dorman and Mazza [15]. We chose MICs as our target calculus with this observation in mind: multirule interaction nets simply lack the expressive power which is necessary to model concurrency in its generality. On the other hand, the solid logical basis and the accompanying type system of differential interaction nets may offer us a more structural way to deal with problems in concurrency theory. The authors believe that studying the nature and structure of the token-flowing can be a way to devise appropriate type structures and logical systems for calculi which lacks any of those, like MICs. This is a topic the authors are currently working on, but which lies outside the scope of this paper.

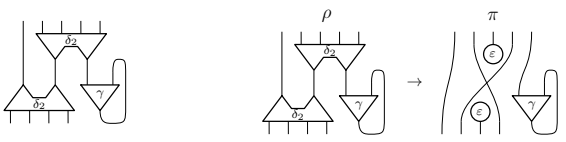
2 Multiport Interaction Combinators at a Glance

This section is devoted to introducing the objects of study of this paper, namely multiport interaction combinators.

2.1 Nets

A *cell* is a triple $(\alpha, \vec{p}, \vec{q})$ of a symbol α, a sequence $\vec{p}$ of *auxiliary ports*, the length of which is the *arity* of α, and a sequence $\vec{q}$ of *principal ports*, the length of which is the *coarity* of α. A cell is drawn in Figure 1(a). A *net* consists of finitely many cells, *free ports* and *wires* which connect each ports with another one. Formally, then, given a set of cell kinds $\mathscr{K}$, a net ρ is $\mathcal{CEL}^\rho$ and $\mathcal{WIR}^\rho$ where:

- $\mathcal{POR}^\rho$ is a finite set of ports.
- $\mathcal{CEL}^\rho$ is a finite set of cells of a kind $\alpha \in \mathscr{K}$, whose ports are elements of $\mathcal{POR}^\rho$.



(a) Net (b) Reduction

Figure 2: An Example

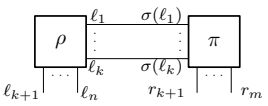


Figure 3: Net Parallel Composition

- $\mathcal{WIR}^\rho$ is a finite set of unordered pairs of ports.
- Each port appears at least once in $\mathcal{WIR}^\rho$ and at most twice in $\mathcal{CEL}^\rho \cup \mathcal{WIR}^\rho$.

The ports of ρ which appear only once in $\mathcal{CEL}^\rho \cup \mathcal{WIR}^\rho$ are said to be *free*. The set of free ports of ρ is $\mathcal{FPOR}^\rho$. Nets are indicated with metavariables like ρ or π. Given a set of cell kinds $\mathscr{K}$, the set of all nets is denoted by $\mathsf{NETS}_{\mathscr{K}}$. We assume that each free port p of any net is labelled with a distinct label ℓ. Often, nets are presented graphically, e.g. a net ρ with free ports labelled with $\ell_1, \ldots, \ell_n$ looks as in Figure 1(b). Metavariables like $\mathcal{L}$ and $\mathcal{M}$ stands for finite sets of locations $\{\ell_1, \ldots, \ell_n\}$.

In *multiport interaction combinators*, one considers cells of three different kinds, namely γ (with arity 2 and coarity 1), δ_n (with arity $2n$ and coarity n, for each $n \geq 1$), and ε (with arity 0 and coarity 1). These are pictured in Figure 1(c). Thus the set of nets in MICs is $\mathsf{NETS}_{\{\gamma,\delta_n,\varepsilon \mid n\geq 1\}}$. A *vicious circle* is a subnet of a net like the one depicted in Figure 1(d), where τ is a tree consisting of γ cells, and ω is a *wiring*, i.e. consists of wires only. An example of an MIC net can be found in Figure 2(a). It consists of two δ_2 cells, one γ cell, and nine free ports, connected in the way depicted. The reason why MICs are a relevant instance of multiport interaction nets is that they allow for a Universality Theorem (see [33], Theorem 6.16, page 209), which states that *any other* system of multiport interaction nets can be encoded into MICs. This makes MICs a minimal, but extremely powerful graph-rewriting formalism, since multiport interaction nets are well known to be expressive enough to faithfully capture the dynamics of process algebras, and of the π-calculus in particular [5, 32]. This is in contrast to multirule interaction nets, like differential interaction nets, which are known to be strictly less expressive than their multiport siblings [15].

Sometimes it is very convenient to form the *parallel composition* of two nets that share certain locations. Any partial injection σ from a set X to a set Y can be seen as a bijection from $dom(\sigma) \subseteq X$ to $rng(\sigma) \subseteq Y$. Its inverse, as usual, is indicated with σ^{-1}. Given nets ρ on $\mathcal{L}$ and π on $\mathcal{M}$ (where $\mathcal{L}$ and $\mathcal{M}$ are disjoint) and a partial injection σ from $\mathcal{L}$ and $\mathcal{M}$, the parallel composition of ρ and π is defined to be the net $\rho \|_\sigma \pi$ depicted in Figure 3, where $\mathcal{L} = \{\ell_1, \ldots, \ell_n\}$ and $\mathcal{M} = \{r_1, \ldots, r_m\}$.

2.2 Reduction

The *interaction rules* for the multiport interaction combinators are the graph rewriting rules (also called *reduction rules*) in Figure 4. We will refer to the three rules as $\gamma\gamma$, $\delta\delta$ and $\gamma\delta$, respectively. A pair of principal ports facing each other in a rule is called a *redex*. Rules can be applied anywhere in a net, giving rise to a binary relation between nets that we indicate as $\rightarrow$.

MICs as defined by Mazza [33] include three more interaction rules, called ε rules, which allow to handle the situation in which an ε cell faces another cell through its principal port. All what we say in this paper holds for the system Mazza considers, and proofs of that can be found in the Appendix. The choice of considering a simplified system is motivated by the desire to make our techniques and results easier to understand, and by the fact that the Universality Theorem holds also in absence of ε rules [12, 33]. For

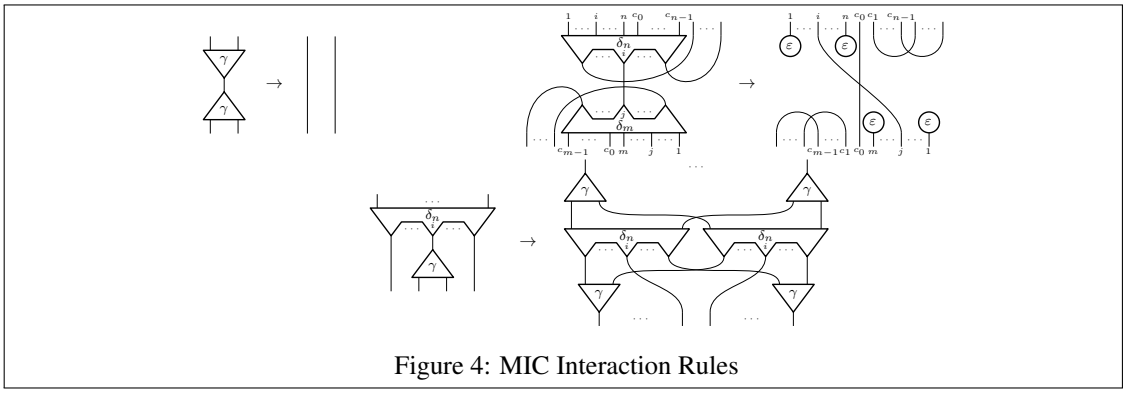


Figure 4: MIC Interaction Rules

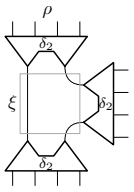


Figure 6: Three δ_2 Cells Facing Each Other

closely related reasons, we decided to consider a system of multiport interaction combinators in which the co-arity n of δ_n cells is always equal to 2 hereafter in the paper, and write NETS for $\mathsf{NETS}_{\{\gamma,\delta_2,\varepsilon\}}$. This, again, allows for a very simple presentation without sacrificing Universality of MICs.

Consider, as an example, the net in Figure 2(a), and call it ρ. It has a redex consisting of two δ_2 cells facing each other through their principal ports. The net ρ can thus be rewritten according to the $\delta\delta$ rule, to another net π, as described in Figure 2(b). The net π has no redex, and is thus said to be in *normal form*. Please observe that a vicious circle occurs in ρ, while no vicious circle can be found in π. This, in other words, is a witness of the fact that vicious circles are *not* preserved by reduction, in general. By the way, we consider vicious circles as a form of a divergent net, given the cyclic chain of γ cells they contain.

2.3 Why One Token is Not Enough

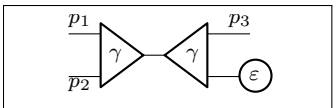


Figure 5: An IC Net.

In the geometry of interaction, at least in its classic incarnation [13, 23], the behaviour of a net is captured by how the net itself transforms a so-called *token* when the latter travels inside it. The net, in other words, is seen as a token transformer, and correctness of the semantics means that reduction turns nets into *equivalent* ones, i.e., into nets which behave the same when seen as token transformers. An example IC net can be found in Figure 5. Seen as a token transformer, the net sends any token which comes from the free port p_1 to p_3; on the other hand any token coming from p_2 gets stuck. This is because the γ cells in the middle work by pushing a symbol on one of the token's stacks when traversed from their secondary to their principal port, and, dually, they pop the symbol when traversed the other way round. The net rewrites to another simple net consisting of an ε cell connected to the free port p_2 and a wire connecting two free ports p_1 and p_3, which (obviously) exhibits the same behaviour seen as a token transformer. This is indeed the way token machines are proved to be a sound model of net reduction in ICs [27].

The same kind of framework cannot work in MICs. To convince yourself about that, please consider the net in Figure 6, and call it ρ. It can reduce to three essentially different nets, namely the ones in Figure 7, call them π_1, π_2, π_3. By analogy with the previous example, it is clear that the way tokens travel inside ρ

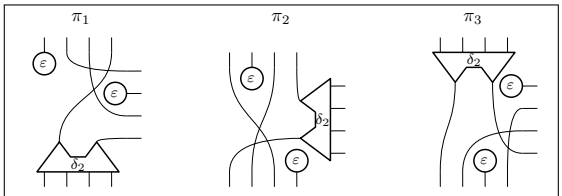


Figure 7: Three Possible Outcomes

should "mimick" the way they travel in *each* of the π_i. But each of the three π_i behaves very differently from the others, i.e., the *nondeterminism* in the choice of which two of the three δ_2 cells in ρ should interact is of a genuine kind. On top of that, there is also the fact that the subnet ξ of ρ could be replaced by a much more complicated net (e.g. that in Figure 2(a)) and thus nondeterministic choices cannot be (all) resolved initially, when the whole computation starts. Some of them need to wait for other choices to be made, have to be *delayed*, and thus naturally become *nonlocal*. One way to deal with nonlocal nondeterminism is to charge tokens the task of resolving it. The inevitable price to pay, however, is the fact that *more than one token* may flow around the net at the same time: there is no way to decide which ones of the many δ_2 cells will be the first ones to meet, and the way out is to be "lazy" and allow *all* δ_2 cells to look for a partner. In ρ, as an example, we would have six tokens, each starting at a principal port of one of the three δ_2 cells. Their task is to look for another cell with which the δ_2 cell they started from could *marry*. Once this is identified, the actual *marriage* can happen, but this must be an indivisible and irreversible operation, after which the same cell cannot marry anyone else. This, in turn, requires the presence of another kind of token, which is static and whose role is to keep track of the "civil state" of any δ_2 cell, i.e. whether it is married or not.

The ones just described are the basic ingredients of the geometry of interaction model we are going to introduce in Section 4 below. Before doing that, however, we need to setup a simple theory of labelled bisimilarity without which one would not even have a way to *define* whether our semantics is correct.

3 Locative Transition Systems and Bisimilarity

Usual, single-token machines, can be seen as token-transformers, i.e., as partial functions capturing the input-output behaviour of a net seen as an automaton turning any token flowing *into* the net into a token flowing *out* of it. This simple picture does not hold anymore in the kind of multi-token machines we work with in this paper. The causal dependencies between tokens are much more complicated here, and we cannot simply proceed by letting token machines to take in input *all* tokens (like in [9, 10]). It is thus natural to see the token machines as labelled transition systems which interact with their environment by letting tokens to flow in and out of them.

It is convenient to introduce a special class of labelled transition systems, called *locative transition systems*, where labels are of a peculiar kind, namely consist of *multisets* of actions played at certain locations. Allowing more than one action to be played together, atomically, in turn allows more than one token to flow in and out at the same time. Locative transition systems support a natural form of parallel composition, and notions of (bi)similarity for which parallel composition can be shown to be a congruence. We will give the status of a locative transition system to multi-token machines in Section 4 below.

3.1 Locative Transition Systems

Action sets, i.e., finite set of actions, are ranged over by metavariables like A and B. For each action set A, we assume the existence of an involution $\mathrm{dual}_A : A \to A$. If for some $a, b \in A$, $\mathrm{dual}_A(a) = b$, we often write $a \perp_A b$ or simply $a \perp b$. For every set X, let $\mathbb{M}(X)$ be the collection of all multisets on X, i.e., of all functions with domain X and codomain $\mathbb{N}$. $\mathbb{FM}(X)$ is the subset of $\mathbb{M}(X)$ consisting of *finite* multisets only. To distinguish between sets and multisets, the latter are denoted with expressions like $\{\!|x_1, \ldots, x_n|\!\}$ instead

of $\{x_1, \ldots, x_n\}$, the latter standing for a set, as usual. Let $\mathcal{L}$ be a finite set of labels, and let A be a set of actions. The expression $\mathcal{L}_A$ stands for the collection $\mathbb{FM}(\mathcal{L} \times A)$. Elements of $\mathcal{L}_A$ are denoted as a, b, etc. Further, if σ is a partial injection on $\mathcal{L}$, a multiset in $\mathbb{FM}(\mathcal{L} \times A)$ is said to be σ-*dual* if it is in the form

$$\{\!\{(\ell_1, a_1), \ldots, (\ell_n, a_n), (\sigma(\ell_1), b_1), \ldots, (\sigma(\ell_n), b_n)\}\!\}$$

where $b_i = \mathrm{dual}_A(a_i)$ for every $1 \leq i \leq n$. The set of σ-dual multisets is $\bot_\sigma$.

We are here interested in labelled transition systems whose labels are drawn from a set in the form $\mathcal{L}_A$. This is said to be a *locative transition system on* $\mathcal{L}_A$ (or a $\mathcal{L}_A$-LTS) and is an ordinary LTS $\mathbf{A} = (S_\mathbf{A}, \rightarrow_\mathbf{A}, s_\mathbf{A})$ where $S_\mathbf{A}$ is a set of *states*, $s_\mathbf{A} \in S_\mathbf{A}$ is the *initial state*, and $\rightarrow_\mathbf{A} \subseteq S_\mathbf{A} \times \mathcal{L}_A \times S_\mathbf{A}$ is the *transition relation*. We write $u \xrightarrow{\mathsf{a}}_\mathbf{A} t$ for $(u, \mathsf{a}, t) \in \rightarrow_\mathbf{A}$.

The parallel composition of two locative transition systems requires a slightly complicated machinery to be defined. Given an $\mathcal{L}_A$-LTS $\mathbf{A}$, an $\mathcal{M}_A$-LTS $\mathbf{B}$ (where $\mathcal{L}$ and $\mathcal{M}$ are disjoint), and a partial injection $\sigma : \mathcal{L} \to \mathcal{M}$, one can define the σ-*parallel composition* $\mathbf{A}\|_\sigma\mathbf{B}$ of $\mathbf{A}$ and $\mathbf{B}$ as the $\mathcal{N}_A$-LTS $(S_\mathbf{A} \times S_\mathbf{B}, \rightsquigarrow, (s_\mathbf{A}, s_\mathbf{B}))$ where $\mathcal{N} = \mathcal{L} \cup \mathcal{M} - dom(\sigma) - rng(\sigma)$, the transition relation $\rightsquigarrow$ is $\rightsquigarrow_\mathbf{A} \cup \rightsquigarrow_\mathbf{B} \cup \rightsquigarrow_\mathbf{AB}$, where

$$\begin{aligned}
\rightsquigarrow_\mathbf{A} &= \{((u_\mathbf{A}, u_\mathbf{B}), \mathsf{a}, (t_\mathbf{A}, u_\mathbf{B})) \mid \mathsf{a} \in (\mathcal{L} - dom(\sigma))_A \wedge u_\mathbf{A} \xrightarrow{\mathsf{a}}_\mathbf{A} t_\mathbf{A}\};\\
\rightsquigarrow_\mathbf{B} &= \{((u_\mathbf{A}, u_\mathbf{B}), \mathsf{a}, (u_\mathbf{A}, t_\mathbf{B})) \mid \mathsf{a} \in (\mathcal{M} - rng(\sigma))_A \wedge u_\mathbf{B} \xrightarrow{\mathsf{a}}_\mathbf{B} t_\mathbf{B}\};\\
\rightsquigarrow_\mathbf{AB} &= \{((u_\mathbf{A}, u_\mathbf{B}), \mathsf{a}, (t_\mathbf{A}, t_\mathbf{B})) \mid \exists \mathsf{d} \in \bot_\sigma . u_\mathbf{A} \xrightarrow{\mathsf{b}}_\mathbf{A} t_\mathbf{A} \wedge u_\mathbf{B} \xrightarrow{\mathsf{c}}_\mathbf{B} t_\mathbf{B}\\
&\qquad \wedge\, \mathsf{a} \cup \mathsf{d} = \mathsf{b} \cup \mathsf{c}\}.
\end{aligned}$$

Informally, $\mathbf{A}\|_\sigma\mathbf{B}$ can perform an action a iff either one of $\mathbf{A}$ or $\mathbf{B}$ can perform it *on one of the non-shared locations*, or both $\mathbf{A}$ and $\mathbf{B}$ perform some actions which, *when put in parallel*, result precisely in a.

Again, please observe that a locative transition system evolves not by performing one (located) action (ℓ, a), but a multiset of such actions.

3.2 Bisimilarity

The notion of (strong) simulation and bisimulation relations on a $\mathcal{L}_A$-LTS can be defined as usual:

- given two such LTSs $\mathbf{A}$ and $\mathbf{B}$, a *simulation* is any subset $\mathcal{R}$ of $S_\mathbf{A} \times S_\mathbf{B}$ such that
  1. $s_\mathbf{A} \mathcal{R} s_\mathbf{B}$;
  2. whenever $u \mathcal{R} t$ and $u \xrightarrow{\mathsf{a}} v$, it holds that $t \xrightarrow{\mathsf{a}} w$, where $v \mathcal{R} w$.
- given two such LTSs $\mathbf{A}$ and $\mathbf{B}$, a *bisimulation* is any subset $\mathcal{R}$ of $S_\mathbf{A} \times S_\mathbf{B}$ such that
  1. $s_\mathbf{A} \mathcal{R} s_\mathbf{B}$;
  2. whenever $u \mathcal{R} t$ and $u \xrightarrow{\mathsf{a}} v$, it holds that $t \xrightarrow{\mathsf{a}} w$, where $v \mathcal{R} w$;
  3. whenever $u \mathcal{R} t$ and $t \xrightarrow{\mathsf{a}} w$, it holds that $u \xrightarrow{\mathsf{a}} v$, where $v \mathcal{R} w$.

  Similarity and bisimilarity are themselves defined as usual:

$$\begin{aligned}
\sim &= \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a bisimulation}\};\\
\precsim &= \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a simulation}\}.
\end{aligned}$$

Weak simulation and bisimulation relations, similarity and bisimilarity on $\mathcal{L}_A$-LTSs are also defined as usual. Given any $\mathcal{L}_A$-LTS, the *weak transition relation* $\Rightarrow$ is defined as

$$\overset{\mathsf{a}}{\Rightarrow} = \begin{cases} \xrightarrow{\emptyset}{}^* \xrightarrow{\mathsf{a}} \xrightarrow{\emptyset}{}^* & \text{if } \mathsf{a} \neq \emptyset \\ \xrightarrow{\emptyset}{}^* & \text{if } \mathsf{a} = \emptyset. \end{cases}$$

- given two such LTSs $\mathbf{A}$ and $\mathbf{B}$, a *weak simulation* is any subset $\mathcal{R}$ of $S_\mathbf{A} \times S_\mathbf{B}$ such that
  1. $s_\mathbf{A} \mathcal{R} s_\mathbf{B}$;
  2. whenever $u \mathcal{R} t$ and $u \xrightarrow{\mathsf{a}} v$, it holds that $t \overset{\mathsf{a}}{\Rightarrow} w$, where $v \mathcal{R} w$.
- given two such LTSs $\mathbf{A}$ and $\mathbf{B}$, a *weak bisimulation* is any subset $\mathcal{R}$ of $S_\mathbf{A} \times S_\mathbf{B}$ such that

1. $s_{\mathbf{A}} \mathcal{R} s_{\mathbf{B}}$;
2. whenever $u \mathcal{R} t$ and $u \xrightarrow{\mathsf{a}} v$, it holds that $t \overset{\mathsf{a}}{\Rightarrow} w$, where $v \mathcal{R} w$;
3. whenever $u \mathcal{R} t$ and $t \xrightarrow{\mathsf{a}} w$, it holds that $u \overset{\mathsf{a}}{\Rightarrow} v$, where $v \mathcal{R} w$.

Weak similarity and bisimilarity are defined as:

$$\approx = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a weak bisimulation}\};$$
$$\lessapprox = \bigcup \{\mathcal{R} \mid \mathcal{R} \text{ is a weak simulation}\}.$$

Noticeably, parallel composition is commutative, modulo strong bisimilarity.

Lemma 1. $\mathbf{A}\|_\sigma \mathbf{B} \sim \mathbf{B}\|_{\sigma^{-1}} \mathbf{A}$.

Proof. We show that

$$\mathcal{R} = \{((u_{\mathbf{A}}, u_{\mathbf{B}}), (u_{\mathbf{B}}, u_{\mathbf{A}})) \mid u_{\mathbf{A}} \in S_{\mathbf{A}}, u_{\mathbf{B}} \in S_{\mathbf{B}}\}$$

is a bisimulation. We have $(s_{\mathbf{A}}, s_{\mathbf{B}}) \mathcal{R} (s_{\mathbf{B}}, s_{\mathbf{A}})$. Assume $(u_{\mathbf{A}}, u_{\mathbf{B}}) \mathcal{R} (u_{\mathbf{B}}, u_{\mathbf{A}})$ and $(u_{\mathbf{A}}, u_{\mathbf{B}}) \xrightarrow{\mathsf{a}} (t_{\mathbf{A}}, t_{\mathbf{B}})$ in $\mathbf{A}\|_\sigma \mathbf{B}$. Let us distinguish three cases:

- If $((u_{\mathbf{A}}, u_{\mathbf{B}}), \mathsf{a}, (t_{\mathbf{A}}, t_{\mathbf{B}})) \in \rightsquigarrow_{\mathbf{A}}$, then $u_{\mathbf{B}} = t_{\mathbf{B}}$ and $u_{\mathbf{A}} \xrightarrow{\mathsf{a}} t_{\mathbf{A}}$. One can see that $\mathbf{B}\|_\sigma \mathbf{A}$ is able to perform $(u_{\mathbf{B}}, u_{\mathbf{A}}) \xrightarrow{\mathsf{a}} (u_{\mathbf{B}}, t_{\mathbf{A}})$ since $((u_{\mathbf{B}}, u_{\mathbf{A}}), \mathsf{a}, (u_{\mathbf{B}}, t_{\mathbf{A}})) \in \rightsquigarrow_{\mathbf{A}}$.
- If $((u_{\mathbf{A}}, u_{\mathbf{B}}), \mathsf{a}, (t_{\mathbf{A}}, t_{\mathbf{B}})) \in \rightsquigarrow_{\mathbf{B}}$, then we can proceed in the similar way.
- If $((u_{\mathbf{A}}, u_{\mathbf{B}}), \mathsf{a}, (t_{\mathbf{A}}, t_{\mathbf{B}})) \in \rightsquigarrow_{\mathbf{AB}}$, then there exist $\mathsf{b} \in \mathcal{L}_A$, $\mathsf{c} \in \mathcal{M}_A$ and $\mathsf{d} \in \perp_\sigma$ such that $u_{\mathbf{A}} \xrightarrow{\mathsf{b}} t_{\mathbf{A}} \wedge u_{\mathbf{B}} \xrightarrow{\mathsf{c}} t_{\mathbf{B}} \wedge \mathsf{a} \cup \mathsf{d} = \mathsf{b} \cup \mathsf{c}$. By definition, σ-dual is equivalent to σ^{-1}-dual, thus we indeed have $((u_{\mathbf{B}}, u_{\mathbf{A}}), \mathsf{a}, (t_{\mathbf{B}}, t_{\mathbf{A}})) \in \rightsquigarrow_{\mathbf{BA}}$.

The opposite direction is similar. □

As expected, strong bisimilarity is a congruence for parallel composition:

Proposition 1. *If* $\mathbf{A} \sim \mathbf{B}$*, then* $\mathbf{A}\|_\sigma \mathbf{C} \sim \mathbf{B}\|_\sigma \mathbf{C}$.

Proof. Let $\mathcal{R}$ be a bisimulation relation between $\mathbf{A}$ and $\mathbf{B}$. We show that

$$\hat{\mathcal{R}} = \{((u_{\mathbf{A}}, u_{\mathbf{C}}), (u_{\mathbf{B}}, u_{\mathbf{C}})) \mid (u_{\mathbf{A}}, u_{\mathbf{B}}) \in \mathcal{R}, u_{\mathbf{C}} \in S_{\mathbf{C}}\}$$

is a bisimulation. We have $(s_{\mathbf{A}}, s_{\mathbf{C}}) \hat{\mathcal{R}} (s_{\mathbf{B}}, s_{\mathbf{C}})$ since $s_{\mathbf{A}} \mathcal{R} s_{\mathbf{B}}$. Assume that $(u_{\mathbf{A}}, u_{\mathbf{C}}) \hat{\mathcal{R}} (u_{\mathbf{B}}, u_{\mathbf{C}})$, and $(u_{\mathbf{A}}, u_{\mathbf{C}}) \xrightarrow{\mathsf{a}} (t_{\mathbf{A}}, t_{\mathbf{C}})$.

- If $((u_{\mathbf{A}}, u_{\mathbf{C}}), \mathsf{a}, (t_{\mathbf{A}}, t_{\mathbf{C}})) \in \rightsquigarrow_{\mathbf{A}}$, then $u_{\mathbf{C}} = t_{\mathbf{C}}$ and $u_{\mathbf{A}} \xrightarrow{\mathsf{a}} t_{\mathbf{A}}$. The latter and $u_{\mathbf{A}} \mathcal{R} u_{\mathbf{B}}$ imply $u_{\mathbf{B}} \xrightarrow{\mathsf{a}} t_{\mathbf{B}}$ and $t_{\mathbf{A}} \mathcal{R} t_{\mathbf{B}}$ for some $t_{\mathbf{B}}$. Therefore $\mathbf{B}\|_\sigma \mathbf{C}$ have the corresponding transition $((u_{\mathbf{B}}, u_{\mathbf{C}}), \mathsf{a}, (t_{\mathbf{B}}, t_{\mathbf{C}})) \in \rightsquigarrow_{\mathbf{A}}$.
- If $((u_{\mathbf{A}}, u_{\mathbf{C}}), \mathsf{a}, (t_{\mathbf{A}}, t_{\mathbf{C}})) \in \rightsquigarrow_{\mathbf{C}}$, then $u_{\mathbf{A}} = t_{\mathbf{A}}$ and $u_{\mathbf{C}} \xrightarrow{\mathsf{a}} t_{\mathbf{C}}$. From the latter, it immediately follows that $(u_{\mathbf{B}}, u_{\mathbf{C}}) \xrightarrow{\mathsf{a}} (u_{\mathbf{B}}, t_{\mathbf{C}})$.
- If $((u_{\mathbf{A}}, u_{\mathbf{C}}), \mathsf{a}, (t_{\mathbf{A}}, t_{\mathbf{C}})) \in \rightsquigarrow_{\mathbf{AC}}$, then there exist $\mathsf{b} \in \mathcal{L}_A$, $\mathsf{c} \in \mathcal{M}_A$ and $\mathsf{d} \in \perp_\sigma$ such that $u_{\mathbf{A}} \xrightarrow{\mathsf{b}} t_{\mathbf{A}} \wedge u_{\mathbf{C}} \xrightarrow{\mathsf{c}} t_{\mathbf{C}} \wedge \mathsf{a} \cup \mathsf{d} = \mathsf{b} \cup \mathsf{c}$. We use $u_{\mathbf{A}} \mathcal{R} u_{\mathbf{B}}$ to obtain $t_{\mathbf{B}}$ such that $u_{\mathbf{B}} \xrightarrow{\mathsf{a}} t_{\mathbf{B}}$ and $t_{\mathbf{A}} \mathcal{R} t_{\mathbf{B}}$. Now observe $((u_{\mathbf{B}}, u_{\mathbf{C}}), \mathsf{a}, (t_{\mathbf{B}}, t_{\mathbf{C}})) \in \rightsquigarrow_{\mathbf{BC}}$.

This concludes the proof. □

The same holds for weak bisimilarity:

Proposition 2. *If* $\mathbf{A} \approx \mathbf{B}$*, then* $\mathbf{A}\|_\sigma \mathbf{C} \approx \mathbf{B}\|_\sigma \mathbf{C}$.

4 Multi-Token Machines

This section is devoted to defining the multi-token machines which are the object of study of this paper, and to proving some basic properties of them. In particular, we will give a Compositionality Theorem that will be very helpful in the following section.

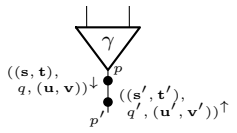


Figure 8: Marriage Tokens: an Example

4.1 States

States of a multi-token machine are just multisets of tokens, each of them consisting of a port in the underlying net, modelling *where* the token is, and of some auxiliary information (e.g. the token's origin, some stacks, etc.), which varies depending on the token's kind. All this will be formalised in this section.

A *stack* $\mathbf{s}$ is any sequence whose elements are either symbols from the alphabet $\{\mathsf{p}, \mathsf{q}\}$ or natural numbers. Formally:

$$\mathbf{s}, \mathbf{t}, \mathbf{u} ::= \epsilon \mid \mathbf{s}\mathsf{p} \mid \mathbf{s}\mathsf{q} \mid \mathbf{s}n,$$

where $n \in \mathbb{N}$ and ϵ is the empty stack. The set of all stacks is $\mathcal{STK}$. A *configuration* is a pair of stacks $(\mathbf{s}, \mathbf{t})$, and is denoted with metavariables like C, D. The set of all configurations is $\mathcal{CON}$. We define the *conjugate* $\overline{\mathbf{s}}$ of a stack $\mathbf{s}$ as follows:

$$\overline{\epsilon} = \epsilon \qquad \overline{\mathbf{s}\mathsf{p}} = \overline{\mathbf{s}}\mathsf{q} \qquad \overline{\mathbf{s}\mathsf{q}} = \overline{\mathbf{s}}\mathsf{p} \qquad \overline{\mathbf{s}n} = \overline{\mathbf{s}}n$$

Given a configuration $C = (\mathbf{s}, \mathbf{t})$, its conjugate $\overline{C}$ is defined to be $(\overline{\mathbf{s}}, \mathbf{t})$.

Throughout this section, wires of the underlying net will be ranged over by metavariables like e, f, g. A *cell type* is any element of the set $\mathcal{CT} = \{\gamma, \delta\}$. Cell types will be denoted with metavariables like c an d. *Tokens* can be of one of four different kinds:

- *Single status tokens*, which are elements of $\mathcal{POR}^\rho \times \mathcal{STK}$. Graphically, single status tokens are denoted with ■.
- *Married status tokens*, which are elements of $\mathcal{POR}^\rho \times \mathcal{STK} \times \mathcal{CT}$. Graphically, married status tokens are denoted with ×.
- *Marriage tokens*, which are elements of $\mathcal{POR}^\rho \times \mathcal{CON} \times \mathcal{POR}^\rho \times \mathcal{CON}$. Graphically, marriage tokens are denoted with ●.
- *Matching tokens*, which are elements of $\mathcal{POR}^\rho \times \mathcal{CON} \times \mathcal{POR}^\rho \times \mathcal{CON}$. Graphically, matching tokens are denoted with ○.

The first components of tokens indicate their current positions. Status tokens must be placed at principal ports while matching tokens must lie at free ports. The second components of marriage and matching tokens are their current configurations. At the third and fourth components, marriage and matching tokens keep their origins and configurations as initially installed. The set of all tokens for the net ρ is indicated as $\mathcal{TKS}^\rho$.

States of the machine we are defining will just be multisets of tokens, and are denoted with metavariables like S and U. We assume that marriage tokens on auxiliary or principal ports of cells are always going out of the cells while those on free ports are coming in from the environment. For example, we depict two marriage tokens $(p, (\mathbf{s}, \mathbf{t}), q, (\mathbf{u}, \mathbf{v}))$ on a principal port p of a γ cell and $(p', (\mathbf{s}', \mathbf{t}'), q', (\mathbf{u}', \mathbf{v}'))$ on a port p' which is connected with p by wire (p' may be either a port of another cell or a free port) as in Figure 8.

The *initial state* of the machine comprises one marriage token $(p, (\epsilon, \epsilon), p, (\epsilon, \epsilon))$ and one single status token (p, ϵ) for each principal port p of each δ_2 cell in the underlying net, as shown in Figure 9. Intuitively, those tokens in the initial state are in charge of keeping track of the status of the δ_2 cell, and to look for possible partners for it.

4.2 Internal Transition Rules

The behaviour of multi-token machines is given by a series of transition rules which prescribe how and when a marriage token can move inside a net, and the protocol governing marriages.

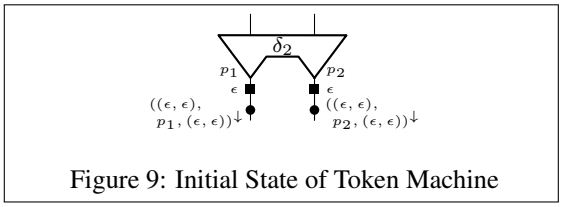


Figure 9: Initial State of Token Machine

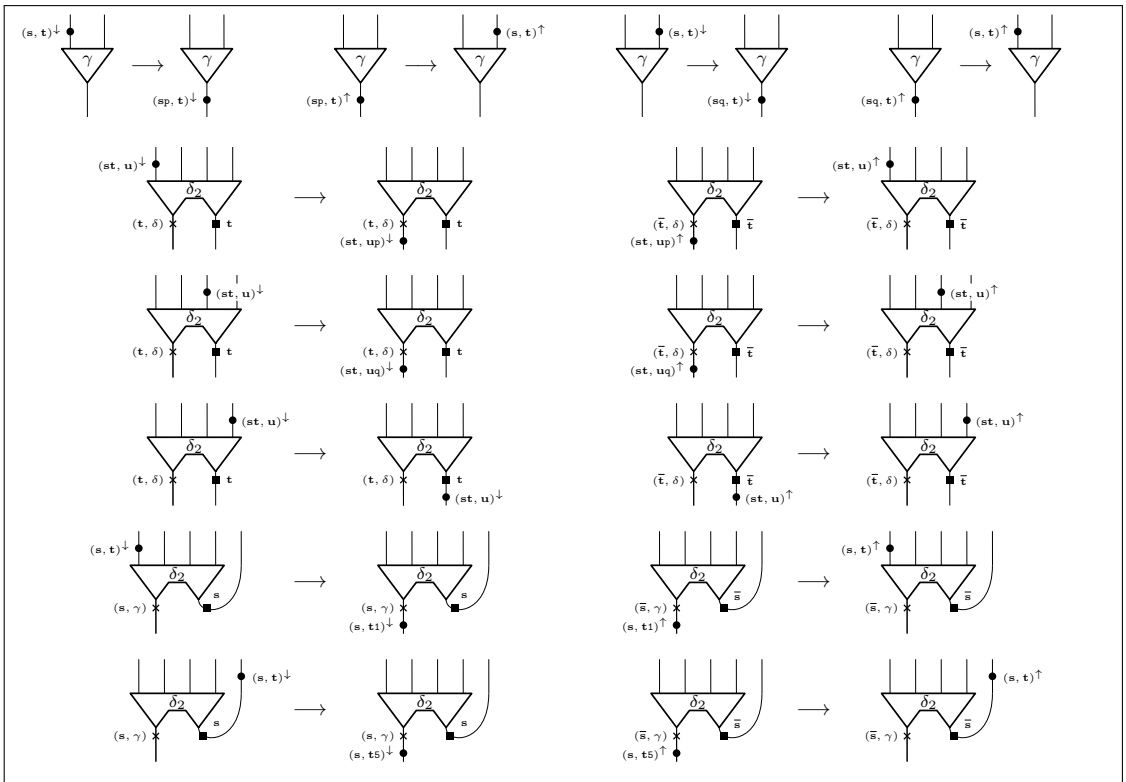


Figure 10: Internal Moving Transition Rules

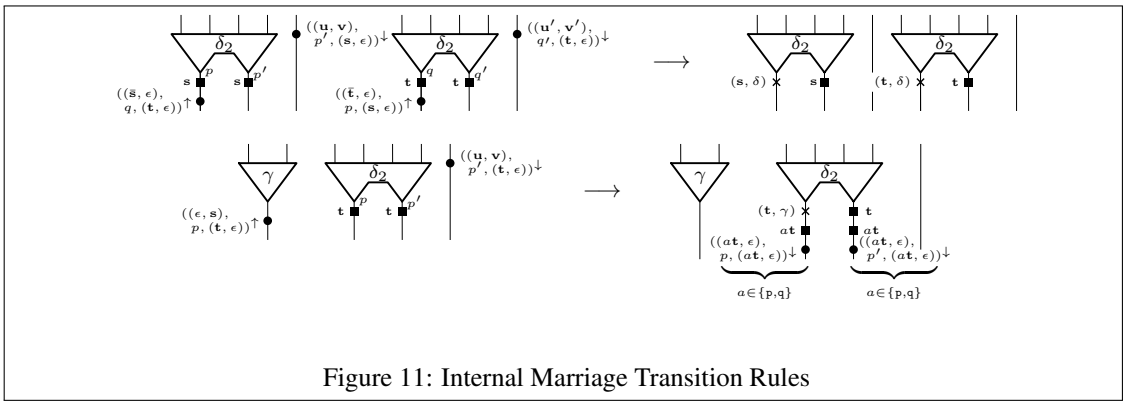


Figure 11: Internal Marriage Transition Rules

Transition rules prescribing how token *moves* are in Figure 10. Please observe how marriage tokens can flow through γ cells more or less the same way they do in interaction combinators [27]: some symbol is either pushed or popped from their first stack. On the other hand, when tokens face a δ_2 cell, there are in principle more than one possibility as for how they should move, all this depending on the presence of certain married status tokens.

As for transition rules performing *marriages*, they are in Figure 11. The first rule handles the marriage between two δ_2 cells, while the second one is charged of handling the case in which a δ_2 cell marries a γ cell. In the first case, it is as if the two δ_2 cells have interacted with each other, thus annihilating themselves, and producing some ε cells. Notice also how the marriage token which originated from *the other* principal port, is annihilated in the process; summing up, there are four tokens involved altogether. In the second case, the δ_2 is virtually duplicated, and indeed some new (status and marriage) tokens are created, each one corresponding to a virtual copy of the δ_2 cell. In both cases, (single and married) status tokens are in charge of guaranteeing atomicity, and of keeping track of whether each (copy of a) δ_2 cell is single or married, in the latter case remembering also the nature of the cell's partner. It is instructive to notice how marriages between two δ_2 cells are bidirectional and symmetric in nature, while those between a δ_2 cell and a γ cell are asymmetric and unidirectional: the latter are inert and marriage tokens are meant to start their journey from δ_2 cells uniquely.

4.3 External Transition Rules

As already pointed out, it is quite convenient to see token machines not simply as automata evolving as described in the previous section, but also as labelled transition systems having a nontrivial *interactive* behaviour. More specifically, token machines can interact with their environment through free ports by inputting or outputting a token, by letting a marriage happen, or by killing a token as a part of a marriage. This makes locative transition systems an ideal candidate for the kind of LTSs one needs here. *External actions*, which express the interactions which could happen at such a location, are generated by the following grammar:

$$A, B ::= \mathsf{out}(C) \mid \mathsf{in}(C) \mid \mathsf{ma}_{cd}(C) \mid \mathsf{kill}(C) \mid \mathsf{cokill}(C)$$

where C ranges over $\mathcal{CON}$ and c, d ranges over $\mathcal{CT}$. The set of external actions is indicated as $\mathcal{EA}$. We define $\mathsf{dual}_{\mathcal{EA}}$ by the following rules:

$$\mathsf{out}(C) \bot \mathsf{in}(C); \qquad \mathsf{ma}_{cd}(C) \bot \mathsf{ma}_{dc}(\overline{C}); \qquad \mathsf{kill}(C) \bot \mathsf{cokill}(C).$$

Actually, any net ρ can be turned into an $\mathcal{FPOR}^{\rho}_{\mathcal{EA}}$-LTS $\mathsf{TM}_{\rho} = (\mathbb{FM}(\mathcal{TKS}^{\rho}), \rightarrow, s_{\rho})$, which is said to be the *token machine for* ρ. This is done considering internal transition rules as producing the empty multiset of action, and by giving some *external* transition rules, namely those in Figure 12 and Figure 13. Multiple external actions involving distinct tokens can be combined in just one labelled transition: this is possible because labels of TM_{ρ} are multisets. Let us briefly comment on the role of each rule:

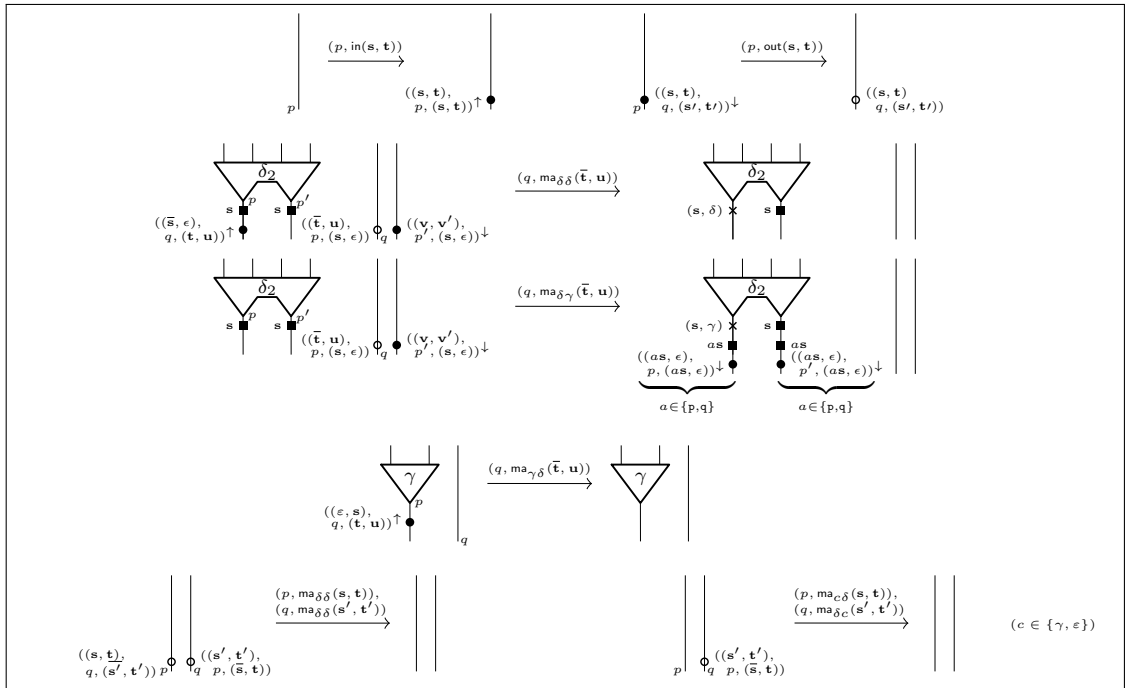


Figure 12: External Transition Rules for Input, Output and Marriages

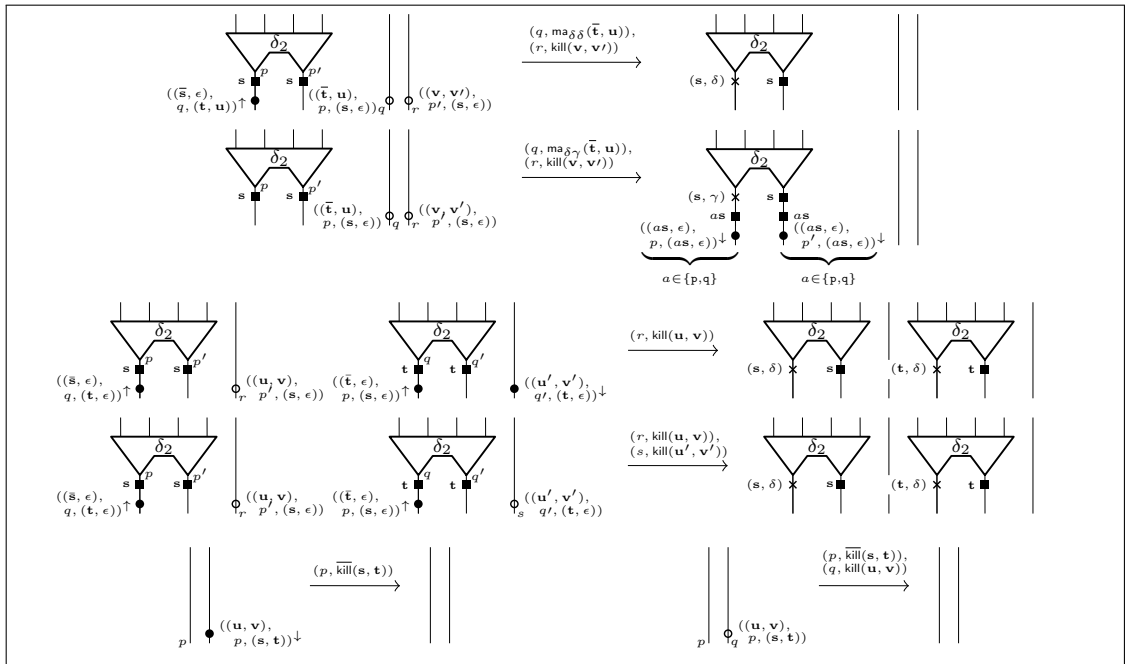


Figure 13: External Transition Rules for Killing Tokens

- The first two rules in Figure 12 allow tokens to flow into the net (thus giving rise to an action $\mathsf{in}(C)$) or to flow out of the net (thus giving rise to an action $\mathsf{out}(C)$). In the latter case, something needs to keep track of the fact that a marriage token has indeed left the net (in a certain configuration): this is precisely the role of matching tokens, that is to say tokens of the fourth (and last) kind.
- The fourth and fifth rules in Figure 12 model the situation in which a δ_2 cell performs a marriage "with the environment", by way of a pair of marriage tokens, the first of which is as usual next to the cell, but the other has flown out the net, a fact witnessed by the presence of a matching token. The fourth rule takes care of the case in which the cell in the environment is itself a δ_2 cell, while the fifth rule accounts for an asymmetric marriage with a γ cell.
- We also have a rule (the third one in Figure 12) that models the marriage between a γ cell and the environment.
- The sixth and seventh rules in Figure 12 forward marriage actions via the matching tokens, emitting dual actions.
- In the third and fourth rules in Figure 12, the marriage token which needs to be killed as a result of the marriage still lies in the net, but nothing guarantees that it has not flown out of the net itself. It is thus necessary to have some further rules, for example the first rule in Figure 13, in which we not only marry a δ_2 cell with the environment but also kill a marriage token which has already flown out of the net, expressed by a $\mathsf{kill}(C)$ action.
- The last two rules in Figure 13 describe how a $\mathsf{kill}(C)$ action is forwarded and how it actually acts on a marriage token, by emitting the dual action $\mathsf{cokill}(C)$.

Having those external rules, two nets can now interact via external actions. The precise situation which it yields will be analysed in the next sections.

4.4 Basic Properties

The expression $\xrightarrow{\emptyset}$ where $\emptyset$ is the empty multiset will be indicated simply as $\to$.

Lemma 2. *Marriage status tokens are inflationary, i.e., if* $\mathsf{S} = \mathsf{U} \cup \{T\}$*, T is a status token, and* $\mathsf{S} \xrightarrow{\ell} \mathsf{V}$*, then* $T \in \mathsf{V}$.

Proof. This is simple case analysis: whenever we perform any internal or external action, status tokens can be created, but once they are part of the current state, they never change their status (and, in particular, they never disappear). □

In the rest of this section, we develop notions and lemmas to show Theorem 1, which will be proved in the next Section 4.5. Suppose we are working with TM_ρ and suppose that $\mathsf{S} \cup \mathsf{U} \xrightarrow{\ell} \mathsf{V}$ where, however, only the tokens from U are allowed to evolve, while those from S cannot change their status. If this is the case, then V is necessarily in the form $\mathsf{S} \cup \mathsf{T}$, and we write $\mathsf{U} \underset{\mathsf{S}}{\xrightarrow{\ell}} \mathsf{T}$. Given a marriage token $T = (p, C, q, D)$ and a pair (r, E) of a port r and a configuration E, a token $T^{OP}_{r,E}$ is defined as $(p, \overline{C}, r, E)$.

Given a token T and multiset of tokens S, an (S, T)-*focused sequence ending in* Y is any sequence Θ of tokens in the form

$$\Theta : X_1 \underset{\mathsf{S}}{\to} X_2 \underset{\mathsf{S}}{\to} \cdots \underset{\mathsf{S}}{\to} X_n,$$

where $X_1 = T$ and $X_n = Y$. The sequence Θ are also called an (S, T)-*focused sequence* or even an S-*sequence*. A state S is said to be *reachable from* U iff there is a sequence of labelled transitions leading U to S:

$$\mathsf{U} \xrightarrow{\ell_1} \cdots \xrightarrow{\ell_n} \mathsf{S}.$$

A state S is said to be *reachable* iff it is reachable from the initial state of the underlying net. A marriage token is said to be an *origin token* iff its origin and current positions coincide. For every marriage token $T = (p, C, q, D)$, *its origin token* is $\mathsf{Orig}(T) = (q, D, q, D)$. For every matching token T, there is a naturally defined marriage token $\mathsf{M2M}(T)$, namely the one that originated T. Given a state S and a marriage token T, T is said to be S-*canonical* iff there is an $(\mathsf{S}, \mathsf{Orig}(T))$-focused sequence ending in T. A matching token T is S-*canonical* if $\mathsf{M2M}(T)$ is. If S is such that for every marriage or matching token T in S, T is

$(\mathsf{S} - \{T\})$-canonical, then S is said to be, simply, *canonical*. Canonicity is preserved by internal or external interaction:

Lemma 3. *If* S *is canonical and* $\mathsf{S} \xrightarrow{\ell} \mathsf{U}$*, then* U *is canonical, too.*

Proof. This can be proved by a simple case analysis on the rule for $\mathsf{S} \xrightarrow{\ell} \mathsf{U}$. □

The way one builds S-focused sequences is essentially a deterministic process:

Lemma 4. *Given two* S*-focused sequences* Θ *and* Ξ*, either* Θ *is a prefix of* Ξ *or vice versa.*

No nondeterminism is involved in building S-focused sequences:

Lemma 5. S*-focused sequences are invertible, i.e. for every such computation*

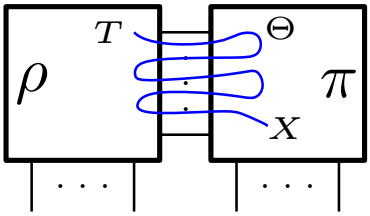
$$\Theta : X_1 \underset{\mathsf{S}}{\rightarrow} X_2 \underset{\mathsf{S}}{\rightarrow} \cdots \underset{\mathsf{S}}{\rightarrow} X_n$$

and for every pair (r, E) *of a port* r *and a configuration* E*, there exists another* S*-focused sequence* $\Theta^{OP}_{r,E}$ *such that*

$$\Theta^{OP}_{r,E} : {X_n}^{OP}_{r,E} \underset{\mathsf{S}}{\rightarrow} {X_{n-1}}^{OP}_{r,E} \underset{\mathsf{S}}{\rightarrow} \cdots \underset{\mathsf{S}}{\rightarrow} {X_1}^{OP}_{r,E}.$$

4.5 Compositionality

Consider any marriage or matching token T in a state $\mathsf{S} \cup \{T\}$ of $\mathsf{TM}_{\rho\|_\sigma\pi}$. If T is S-canonical, then we can trace back T (or $\mathsf{M2M}(T)$) to an origin token X by way of an S-focused sequence. Lemma 5 tells us that from X we can go back to T (or $\mathsf{M2M}(T)$), again by way of an S-focused sequence, that we call Θ. Intuitively, Θ is the computation that brought T where it is now. We can see it as a path in $\rho\|_\sigma\pi$, and depict it graphically as follows:



In particular, Θ can cross the border between ρ and π several times. If we want to somehow simulate Θ in $\mathsf{TM}_\rho\|_\sigma\mathsf{TM}_\pi$, we need to keep track of each crossing by a matching token lying at the border, on the side of the sub-net we are leaving. Let U the set of all these matching tokens which are tokens of either TM_ρ or of TM_π. What we have just implicitly defined is a pair of states $g_\rho(\mathsf{S}, T)$ and $g_\pi(\mathsf{S}, T)$: they are defined as the portions of $\mathsf{S} \cup \mathsf{U}$ which are tokens of TM_ρ and TM_π, respectively. If T is not S-canonical, then

Theorem 1 (Compositionality)**.** $\mathsf{TM}_{\rho\|_\sigma\pi} \approx (\mathsf{TM}_\rho)\|_\sigma(\mathsf{TM}_\pi)$.

Proof. We need to define a binary relation $\mathcal{R}$ between the states of the two involved LTSs, and then prove it to be a bisimulation. The simplest way to describe $\mathcal{R}$ is as the set of all the pairs in the form $(\mathsf{S}, f^*(\mathsf{S}))$, where S is a canonical state of $\rho\|_\sigma\pi$, and f^* is defined as

$$f^*(\mathsf{S}) = (\cup_{T\in\mathsf{S}} f_\rho(\mathsf{S}, T), \cup_{T\in\mathsf{S}} f_\pi(\mathsf{S}, T))$$

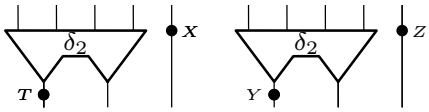
and the partial function f_ρ (respectively, f_π) maps a pair in the form (S, T) into a state of TM_ρ (respectively, of TM_π). The way f_ρ and f_π are defined depends on the nature of T. In particular,

- If T is either a single status token or a marriage status token lying next to its natural position, say a cell in ρ, then $f_\rho(\mathsf{S}, T)$ will be just $\{X\}$, where X is the counterpart of T in ρ, while $f_\pi(\mathsf{S}, T)$ will be just $\emptyset$. Similarly when the position of T is a cell in π. If T is not at a natural position, then both f_ρ an f_π are undefined at (S, T).

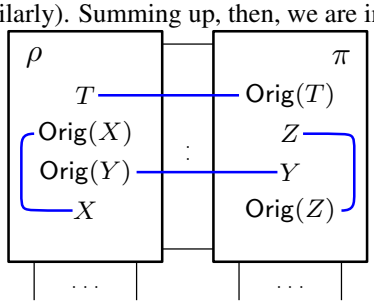
- If T is a marriage or matching token, then $f_\rho(\mathsf{S}, T) = g_\rho(\mathsf{S}, T)$ and $f_\pi(\mathsf{S}, T) = g_\pi(\mathsf{S}, T)$ where g_ρ and g_π are the maps defined above.

We now need to prove that $\mathcal{R}$ is a bisimulation relation. To do that, we need to prove three statements:

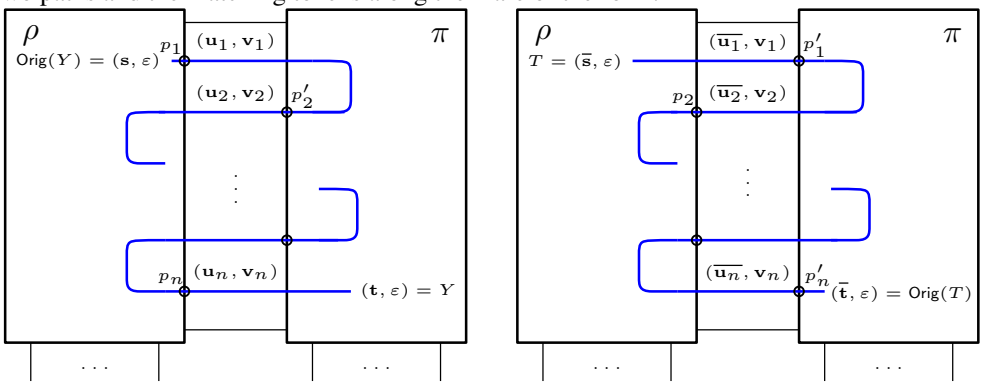
1. We first of all need to show that $(s_{\rho\|_\sigma\pi}, (s_\rho, s_\pi)) \in \mathcal{R}$. To prove that, first of all observe that $s_{\rho\|_\sigma\pi}$ does not contain any matching token, that all the marriage tokens in it are trivially canonical, and are thus mapped simply to "themselves" by f^*.
2. We then need to show that if $(\mathsf{S}, \mathsf{U}) \in \mathcal{R}$ and $\mathsf{S} \xrightarrow{\ell} \mathsf{V}$, then $\mathsf{U} \overset{\ell}{\Rightarrow} \mathsf{T}$ where $(\mathsf{V}, \mathsf{T}) \in \mathcal{R}$. We need to analyse several cases:
   - If V is obtained from S by performing a crossing internal transition, then the crossing can be easily simulated in U, the only proviso being that if the involved marriage token in S lies at the border between ρ and π, then the simulating transitions could be *two* instead of *one*, namely a crossing and matching input-output pair.
   - If V is obtained from S by performing an internal $\delta\delta$ marriage, then the four involved marriage tokens are as follows



   where X and Y originated in the leftmost cell, while T and Z originated in the rightmost cell. The four tokens are canonical, and are thus mapped by f^* to four marriage tokens, possibly with some matching tokens lying along the paths leading each of them to its origin. Now, suppose, just as an example, that the leftmost cell and link above are in ρ, while rightmost cell and link are in π (the other cases can be handled similarly). Summing up, then, we are in the following situation:

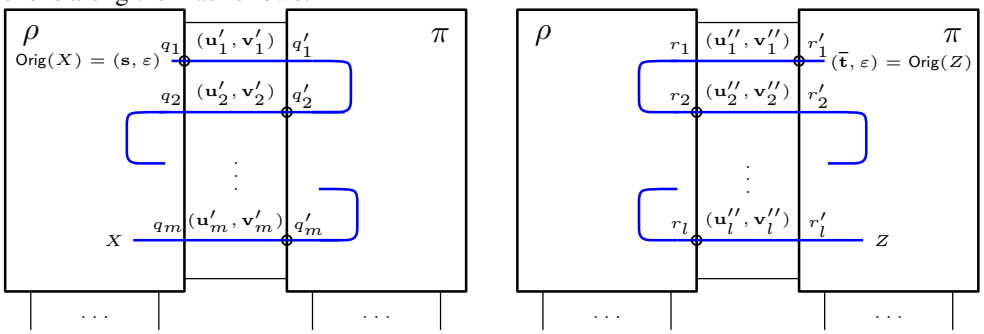


   where the four depicted paths can of course, in principle, go back and forth between ρ and π, even several times. Now, observe that:
   - Along the four paths, appropriate matching tokens in $f^*(\mathsf{S})$ mark the crossing of the border between ρ and π.
   - Let $T = (q, \overline{C}, p, D)$ and $Y = (p, \overline{D}, q, C)$ (by definition of the initial state and transition rules they are necessarily in this form). $T^{OP}_{q,C} = \mathsf{Orig}(Y)$ and $Y^{OP}_{p,D} = \mathsf{Orig}(T)$, and by Lemma 5, there exist paths from $\mathsf{Orig}(T)$ to T and to $\mathsf{Orig}(Y)$ to Y that are the inverses of each other. Then, the two paths and the matching tokens along them are of the form:



   where $\sigma(p_i) = p'_i$ and $n = 2n' + 1$ for some $n' \geq 0$.
   - The paths from $\mathsf{Orig}(X)$ to X and from $\mathsf{Orig}(Z)$ to Z are depicted together with the matching

tokens along them as follows:

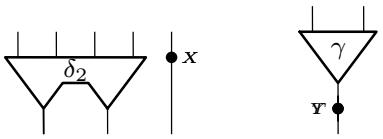


where $\sigma(q_i) = q'_i$, $\sigma(r_i) = r'_i$, $m = 2m'$ and $l = 2l'$ for $m', l' \geq 0$.

As a consequence, one can prove that U silently evolves to a state T. More specifically, if $\mathsf{U} = (\mathsf{U}_\rho, \mathsf{U}_\pi)$, then:

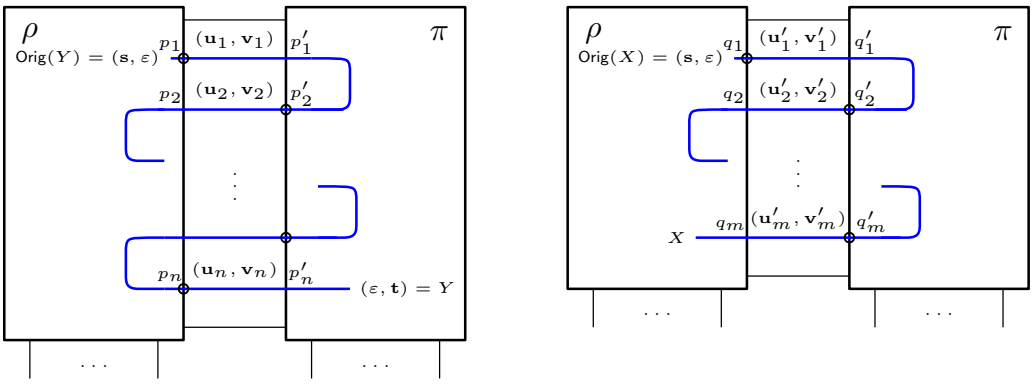
- U_ρ performs an external $\delta\delta$ marriage and several marriages between free ports, triggering a bag of actions a_ρ consisting of:
  - $(p_1, \mathsf{ma}_{\delta\delta}(\mathbf{u}_1, \mathbf{v}_1))$
  - $(p_{2i}, \mathsf{ma}_{\delta\delta}(\overline{\mathbf{u}_{2i}}, \mathbf{v}_{2i}))$ and $(p_{2i+1}, \mathsf{ma}_{\delta\delta}(\mathbf{u}_{2i+1}, \mathbf{v}_{2i+1}))$ for each $i = 1, \ldots, n'$
  - $(q_{2i-1}, \mathsf{kill}(\mathbf{u}'_{2i-1}, \mathbf{v}'_{2i-1}))$ and $(q_{2i}, \mathsf{cokill}(\mathbf{u}'_{2i}, \mathbf{v}'_{2i}))$ for each $i = 1, \ldots, m'$
  - $(r_{2i-1}, \mathsf{cokill}(\mathbf{u}'_{2i-1}, \mathbf{v}'_{2i-1}))$ and $(r_{2i}, \mathsf{kill}(\mathbf{u}'_{2i}, \mathbf{v}'_{2i}))$ for each $i = 1, \ldots, l'$
- U_π performs an external $\delta\delta$ marriage and marriages between free ports which invoke a_π consisting of:
  - $(p'_{2i-1}, \mathsf{ma}_{\delta\delta}(\overline{\mathbf{u}_{2i-1}}, \mathbf{v}_{2i-1}))$ and $(p'_{2i}, \mathsf{ma}_{\delta\delta}(\mathbf{u}_{2i}, \mathbf{v}_{2i}))$ for each $i = 1, \ldots, n'$
  - $(p'_n, \mathsf{ma}_{\delta\delta}(\mathbf{u}_n, \mathbf{v}_n))$
  - $(q'_{2i-1}, \mathsf{cokill}(\mathbf{u}'_{2i-1}, \mathbf{v}'_{2i-1}))$ and $(q'_{2i}, \mathsf{kill}(\mathbf{u}'_{2i}, \mathbf{v}'_{2i}))$ for each $i = 1, \ldots, m'$
  - $(r'_{2i-1}, \mathsf{kill}(\mathbf{u}'_{2i-1}, \mathbf{v}'_{2i-1}))$ and $(r'_{2i}, \mathsf{cokill}(\mathbf{u}'_{2i}, \mathbf{v}'_{2i}))$ for each $i = 1, \ldots, l'$
- We can see they are balanced, i.e., $\mathsf{a}_\rho \cup \mathsf{a}_\pi \in \perp_\sigma$. Thus U can perform these marriages in an internal transition. The resulting state T coincides with $f^*(\mathsf{V})$.

- If V is obtained from S by performing an internal $\gamma\delta$ marriage, then we have two marriage tokens involved:



Both of the tokens X and Y originated in the δ_2 cell. Similarly to the case of $\delta\delta$ marriage, the marriage can be internally simulated in $(\mathsf{TM}_\rho) \|_\sigma (\mathsf{TM}_\pi)$ via finitely many communication between TM_ρ and TM_π:

- The paths of T and X crosses the border finitely many times, for example as in the following figure, where $\sigma(p_i) = p'_i$, $n = 2n' + 1$ for some $n' \geq 0$, and $m = 2m'$ for some $m' \geq 0$. .



- A bag of actions U_ρ in a_ρ consisting of

- $(p_1, \mathsf{ma}_{\delta\gamma}(\mathbf{u}_1, \mathbf{v}_1))$
- $(p_{2i}, \mathsf{ma}_{\gamma\delta}(\overline{\mathbf{u}_{2i}}, \mathbf{v}_{2i}))$ and $(p_{2i+1}, \mathsf{ma}_{\delta\gamma}(\mathbf{u}_{2i+1}, \mathbf{v}_{2i+1}))$ for each $i = 1, \ldots, n'$
- $(q_{2i-1}, \mathsf{kill}(\mathbf{u}'_{2i-1}, \mathbf{v}'_{2i-1}))$ and $(q_{2i}, \mathsf{cokill}(\mathbf{u}'_{2i}, \mathbf{v}'_{2i}))$ for each $i = 1, \ldots, m'$

and another bag of actions U_π in a_π consisting of

- $(p_{2i}, \mathsf{ma}_{\delta\gamma}(\mathbf{u}_{2i}, \mathbf{v}_{2i}))$ and $(p_{2i-1}, \mathsf{ma}_{\gamma\delta}(\overline{\mathbf{u}_{2i-1}}, \mathbf{v}_{2i-1}))$ for each $i = 1, \ldots, n'$
- $(p_n, \mathsf{ma}_{\gamma\delta}(\overline{\mathbf{u}_1}, \mathbf{v}_1))$
- $(q_{2i-1}, \mathsf{cokill}(\mathbf{u}'_{2i-1}, \mathbf{v}'_{2i-1}))$ and $(q_{2i}, \mathsf{kill}(\mathbf{u}'_{2i}, \mathbf{v}'_{2i}))$ for each $i = 1, \ldots, m'$

allow us to perform an internal marriage that yields a state $\mathsf{T} = f^*(\mathsf{V})$.

- If V is obtained from S by performing any other internal transition, it must be an internal transition on a marriage token that does not cross the border. We can easily obtain $\mathsf{T} = f^*(\mathsf{V})$ by performing an internal transition on the corresponding marriage token in the same side (either in TM_ρ or TM_π) of the token machine.
- If V is obtained from S by performing input action $\mathsf{in}(C)$, it is again easily simulated in $(\mathsf{TM}_\rho)\|_\sigma(\mathsf{TM}_\pi)$ by performing the same input action on the same free port.
- If V is obtained from S by performing output action $\mathsf{out}(C)$ on a token with stacks C, there must be a token with the same stack C on the same free port in $(\mathsf{TM}_\rho)\|_\sigma(\mathsf{TM}_\pi)$. We can simulate the action by performing output action on that token.
- The other cases (external marriages and killing/killed) can be simulated similarly to the internal marriage cases: a corresponding action and several times of communication between TM_ρ and TM_π yield the state $\mathsf{T} = f^*(\mathsf{V})$.
- The cases above extend to any external transition with multiple actions. This is because of the following reason. For each action, the involved tokens have their path from its origin as discussed above. Those paths (and matching tokens along them) are all distinct and thus their corresponding transitions in $(\mathsf{TM}_\rho)\|_\sigma(\mathsf{TM}_\pi)$ do not interfere with each other.

3. We then need to show that if $(\mathsf{S}, \mathsf{U}) \in \mathcal{R}$ and $\mathsf{U} \xrightarrow{\ell} \mathsf{V}$, then $\mathsf{S} \overset{\ell}{\Rightarrow} \mathsf{T}$ where $(\mathsf{T}, \mathsf{V}) \in \mathcal{R}$. By definition, the transition relation $\to$ on $(\mathsf{TM}_\rho)\|_\sigma(\mathsf{TM}_\pi)$ is the union of three relations $\to_\rho$, $\to_\pi$ and $\to_{\rho\pi}$. Now, if $\mathsf{U} \xrightarrow{\ell}_\rho \mathsf{V}$ or $\mathsf{U} \xrightarrow{\ell}_\pi \mathsf{V}$, then finding T is trivial. If, on the other hand, $\mathsf{U} \xrightarrow{\ell}_{\rho\pi} \mathsf{V}$, then by definition,

$$\begin{gathered}\mathsf{U} = (\mathsf{U}_\rho, \mathsf{U}_\pi);\\ \mathsf{V} = (\mathsf{V}_\rho, \mathsf{V}_\pi);\\ \ell \cup \xi = \nu_\rho \cup \nu_\pi;\\ \mathsf{U}_\rho \xrightarrow{\nu_\rho} \mathsf{V}_\rho;\\ \mathsf{U}_\pi \xrightarrow{\nu_\pi} \mathsf{V}_\pi.\end{gathered}$$

Again, if $\xi = \emptyset$, then finding T with the required property is trivial because $\mathsf{S} \xrightarrow{\nu_\rho} \mathsf{T}$ and $\mathsf{S} \xrightarrow{\nu_\pi} \mathsf{T}$, then $\mathsf{S} \xrightarrow{\ell} \mathsf{T}$, since $\ell = \nu_\rho \cup \nu_\pi$. If ξ is *not* empty, then we need to analyse several cases, keeping in mind that ξ is by definition of parallel composition a σ-dual multiset:

- Suppose that $(\ell, \mathsf{out}(C)) \in \xi$. Then, $(\sigma(\ell), \mathsf{in}(C))$ is also an element of ξ. This can be simulated in S by just an internal transition. Similarly if $(\ell, \mathsf{in}(C)) \in \xi$.
- Suppose that $(\ell, \mathsf{ma}_{\gamma\delta}(C)) \in \xi$ as an effect of an external marriage between a γ cell and a free port. Then there must also be another action $(\sigma(\ell), \mathsf{ma}_{\delta\gamma}(\overline{C}))$ in ξ. We can think of this action as being produced by a (possibly empty) chain of marriages between two free ports alternating between ρ and π. The fact that the chain alternates is again due to σ-duality of ξ. Let us distinguish a few further cases depending on the rule in which this chain ends:
  - If the chain ends in an *external* marriage between a δ cell and a free port, then the whole chain, including its extremes, can be simulated by just an *internal* $\gamma\delta$ marriage. If the latter comes in its version with a killing action in the form $(\ell', \mathsf{kill}(C))$, then we need to do the same kind of chain-based reasoning.
  - If the chain ends in a marriage between two free ports one of the two not being in the domain of σ, then the whole chain, including its extremes, can be simulated by an *external* marriage between a γ cell and a free port.

- Suppose that $(\ell, \mathsf{ma}_{\gamma\delta}(C)) \in \xi$ as the effect of an external marriage between an δ and a free port. Then, again, we can proceed as in the last but previous case, with an additional complication due to the presence of killing actions at *both* extremes of the chain.
- Suppose that $(\ell, \mathsf{ma}_{\gamma\delta}(C)) \in \xi$ as the effect of an external marriage between two free ports. Then we can proceed with the usual chain-based reasoning, but on *both* sides of the action we have just identified.

This way, we can "strip-off" all the actions from ξ and ℓ, and each of the chains of actions identified above corresponds to an action in $\mathsf{TM}_{\rho\|_\pi\sigma}$.

□

While the Compositionality Theorem holds in single-token machines as well [13], it has not been considered in existing work on multi-token machines [9, 10], partially because the kind of nets to which the theory can be applied, the so-called *closed nets*, is quite restricted.

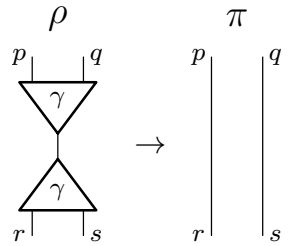
5 Soundness

As already mentioned several times, the transition rules for our token machines are tailored for capturing the behaviours of interaction rules of MICs. This intention is unsurprisingly formalised as a bisimulation between the machine for a net containing a redex (say ρ) and the machine for the net containing its reduct (say π). This, of course, must be done for each interaction rule. However, there is a subtlety. While the result for $\gamma\gamma$ rule is a mere bisimulation of the two machines TM_ρ and TM_π, those for the rules involving δ_2 cells (i.e., $\gamma\delta$ and $\delta\delta$ rules) are not; instead, the results are described as bisimulations between the machines TM_π and TM_ρ in which the initial state is replaced by a state *after a specific marriage transition*. At the level of the entire machines TM_ρ and TM_π, the relations are thus simulations rather than bisimulations. This is because the token machine for a net is designed to simulate *any* possible sequence of reductions from it, while the reduction under consideration is *a particular one* of those all possibilities, since reduction is nondeterministic. Therefore, a reduction on a δ_2 cell always results in discarding some otherwise possible behaviours. This is faithfully handled by a marriage transition in the token machine TM_ρ corresponding to that reduction. In the following three lemmas, we study how TM_ρ and TM_π relate whenever $\rho \to \pi$. This is done by giving certain relations, that we indicate with $\mathcal{R}^\rho_\pi$, between the states of TM_ρ and those of TM_π. Those relations will be useful also in Section 6 below.

Let us first consider $\gamma\gamma$ reduction. In this case, the machines for the redex and the reduct behave precisely the same:

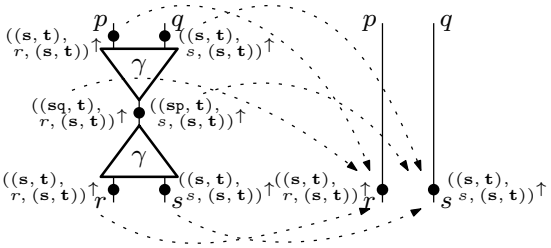
Lemma 6. *If $\rho \to_{\gamma\gamma} \pi$, then $\mathsf{TM}_\rho \approx \mathsf{TM}_\pi$.*

Proof. If $\rho \to_{\gamma\gamma} \pi$, then we are in the following situation:



where p, q, r, s indicate the free ports of ρ and π. We have a partial function $f : \mathcal{TKS}^\rho \rightharpoonup \mathcal{TKS}^\pi$ such that:

- Matching tokens are mapped to the same position in π if their origins are free ports.
- Marriage tokens are pushed back to their origins in π if they are $\emptyset$-canonical. Pictorially, it can be illustrated as:



and similarly for downward tokens.

The relation $\mathcal{R}$ we are looking for is defined as a pointwise extension of f. We can now show that, indeed, $\mathcal{R}$ is a bisimulation relation.

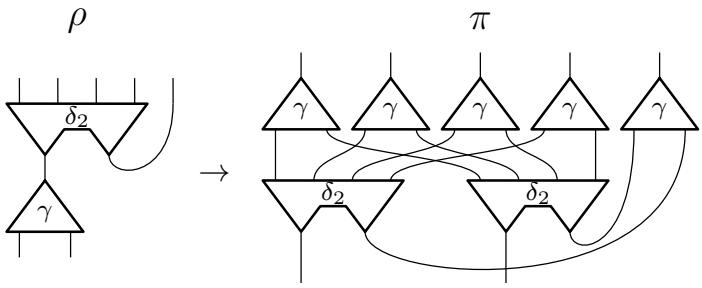
- Both s_ρ and s_π are empty. By definition, $\mathcal{R}$ relates s_ρ and s_π.
- Assume $\mathsf{S}_\rho \ \mathcal{R} \ \mathsf{S}_\pi$ and $\mathsf{S}_\rho \xrightarrow{\mathsf{a}} \mathsf{U}_\rho$. We need to find $\mathsf{S}_\pi \stackrel{\mathsf{a}}{\Rightarrow} \mathsf{U}_\pi$ such that $\mathsf{U}_\rho \ \mathcal{R} \ \mathsf{U}_\pi$. Let us proceed by analysing the three transition types.
  - If $\mathsf{a} = \emptyset$, then the transition must be an internal move of a marriage token because internal marriages never happen in ρ. The definition of f tells us that $\mathsf{U}_\rho \ \mathcal{R} \ \mathsf{S}_\pi$, so we can take $\mathsf{U}_\pi = \mathsf{S}_\pi$.
  - If a is an input action, let $\mathsf{U}_\rho = \mathsf{S}_\rho \uplus \{\!|T|\!\}$ where T is the incoming token. Observe that S_π can evolve into $\mathsf{U}_\pi = \mathsf{S}_\pi \uplus \{\!|T|\!\}$ by performing the same input action and $\mathsf{U}_\rho \ \mathcal{R} \ \mathsf{U}_\pi$.
  - If a is an output action, let $\mathsf{U}_\rho = \mathsf{S}_\rho \setminus \{\!|T|\!\} \uplus \{\!|X|\!\}$ where T is the outgoing token and X is a matching token. We can see that $f(T)$ is also ready to flow out and hence $\mathsf{S}_\pi \xrightarrow{\mathsf{a}} \mathsf{S}_\pi \setminus \{\!|f(T)|\!\} \uplus \{\!|X|\!\}$.
  - If a is a bunch of external marriages, then they should be marriages between free ports because no marriage tokens in S_ρ are placed on e with empty γ-stacks. These actions come from matching tokens in S_ρ and let V be such tokens. They are included in S_π as well. Thus we have $\mathsf{S}_\pi \xrightarrow{\mathsf{a}} \mathsf{S}_\pi \setminus \mathsf{V}$.
- Assume $\mathsf{S}_\rho \ \mathcal{R} \ \mathsf{S}_\pi$ and $\mathsf{S}_\pi \xrightarrow{\mathsf{a}} \mathsf{U}_\pi$. As π doesn't allow any internal actions, we consider external actions.
  - If a is an input, then we can find the counterpart in ρ as we did before.
  - If a is an output, we can suppose, without loss of generality, a marriage token $T = (r, (\mathbf{s}, \mathbf{t}), r, (\mathbf{s}, \mathbf{t})) \in \mathsf{S}_\pi$ flows out through p. Then S_ρ has a marriage token which belongs to $f^{-1}(T)$. The token can reach p with a configuration $(\mathbf{s}, \mathbf{t})$ after a few internal transitions and its origin is the same as T. So it allows TM_ρ to perform the compatible output action.
  - If a consists of marriages, TM_ρ is able to consume the corresponding matching tokens in S_ρ.

□

If one of the cells involved in a reduction is a δ_2 cell, one cannot hope to get a result as strong as Lemma 6:

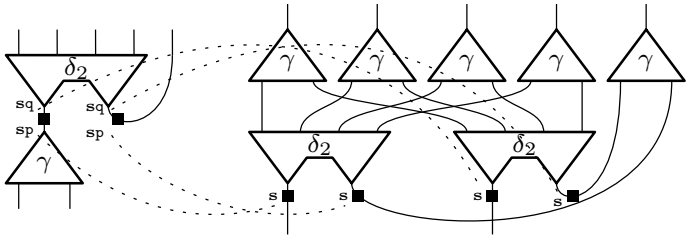
Lemma 7. *If $\rho \ \rightarrow_{\gamma\delta} \ \pi$, then there is t_ρ such that $s_\rho \ \rightarrow \ t_\rho$ by an internal marriage transition and $(\mathbb{FM}(\mathcal{TKS}^\rho), \rightarrow, t_\rho) \approx \mathsf{TM}_\pi$.*

Proof. We are in the following situation:



Observe that s_ρ can perform a marriage between the γ and δ_2 cells, resulting in a state t_ρ which has 1 married status tokens, 5 single status tokens and 4 marriage tokens. We denote the 2 status tokens in t_ρ which have empty γ stacks by V. We define a relation $\mathcal{R}$ between $\mathbb{FM}(\mathcal{TKS}^\rho)$ and $\mathbb{FM}(\mathcal{TKS}^\pi)$, which is described in a pointwise manner as follows:

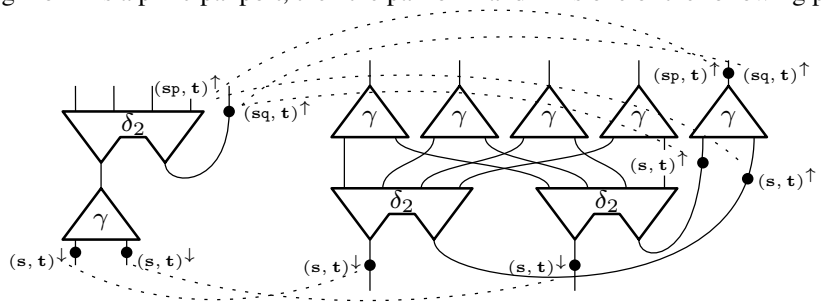
- Status tokens in ρ whose topmost symbols are $\mathtt{p}$ (resp. $\mathtt{q}$) are related to ones of the δ cell on the left (resp. on the right) in π:



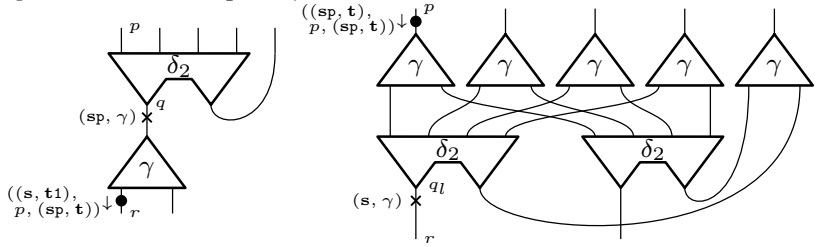
  and similarly for married status tokens.
- Marriage tokens originating from principal ports are related if their origins are related in a similar way to the one described above. Marriage tokens originating from free ports are related if they have the same origins.
- Matching tokens are related similarly to marriage tokens.

Then let $\mathcal{R}' = \{(\mathsf{S}_\rho, \mathsf{S}_\pi) \mid (\mathsf{S}_\rho - \mathsf{V}, \mathsf{S}_\pi) \in \mathcal{R}, \mathsf{S}_\rho \text{ and } \mathsf{S}_\pi \text{ are canonical}\}$. We claim $\mathcal{R}'$ is a bisimulation.

- We can see that $\mathcal{R}'$ relates t_ρ and s_π.
- Suppose $(\mathsf{S}_\rho, \mathsf{S}_\pi) \in \mathcal{R}'$ and $\mathsf{S}_\rho \xrightarrow{\mathsf{a}} \mathsf{U}_\rho$.
  - If $\mathsf{a} = \emptyset$, then one of marriage tokens in S_ρ moved. But the marriage token and its counterpart in S_π are still related and so S_π can remain unchanged.
  - If a is an input action, it is easy to see that making the same input in π will lead S_π to a state related to U_ρ.
  - If a is an output action, let T be the outgoing token in S_ρ and T' be the token in S_π related to T. There are two cases.
    - If the origin of T is a principal port, then the pair of T and T' is one of the following pairs:



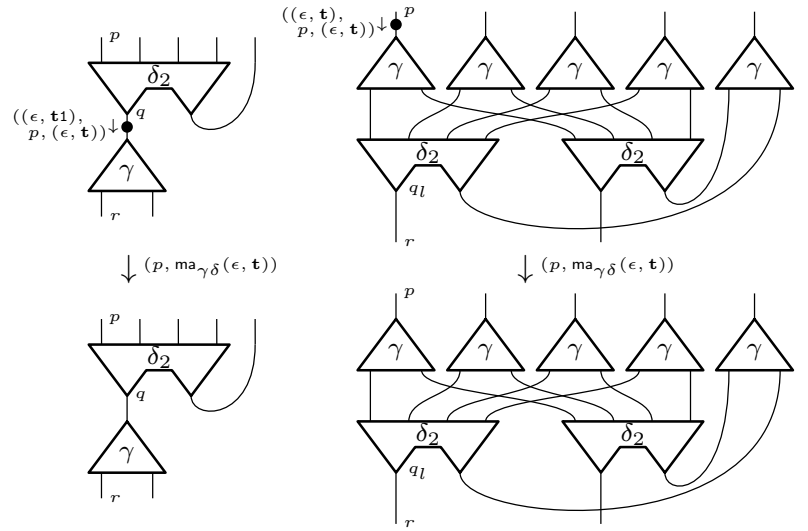
      In any case, T' can reach the same position and configuration as T within zero or one transition steps. Thus we have $\mathsf{S}_\pi \overset{\mathsf{a}}{\Rightarrow} \mathsf{U}_\pi$.
    - If the origin of T is a free port, then T' is on this free port. By definition of $\mathcal{R}'$, S_ρ has a status tokens Y which is necessary for T to make a trip from its origin to its current position. The counterpart Y' for Y is contained in S_π and indeed helps T' to go to the correct location. For instance, let us consider the situation below, in which we have $T = (r, (\mathbf{s}, \mathbf{t}1), p, (\mathbf{sp}, \mathbf{t}))$, $T' = (r, (\mathbf{sp}, \mathbf{t}), p, (\mathbf{sp}, \mathbf{t}))$ and $Y = (q, (\mathbf{sp}, \gamma))$. Notice that T' can arrive at r thanks to $Y' = (q_l, (\mathbf{s}, \gamma))$.



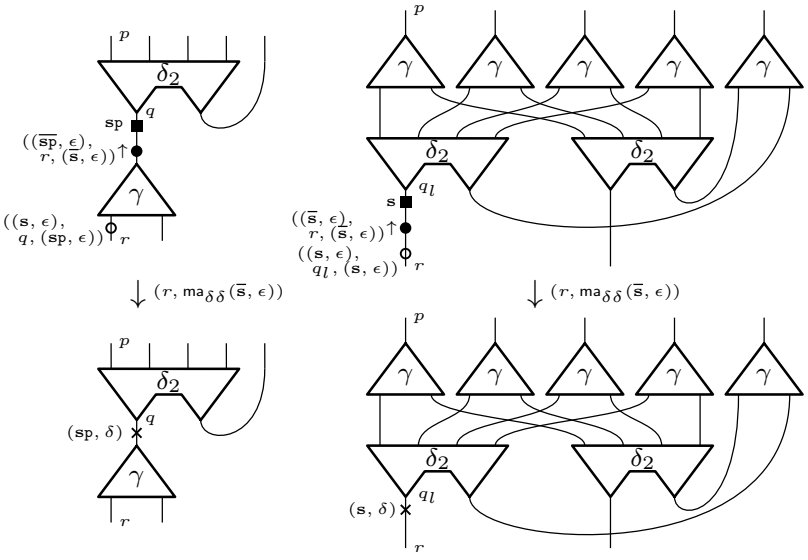
- If a is a multiset of external marriages, each element of a is either a marriage between a principal port and a free port or one between two free ports.
  - A marriage between two free ports involves one or two matching tokens whose origins are free ports. Since such matching tokens in S_ρ are related to their copies in S_π, S_π can imitate that

marriage.

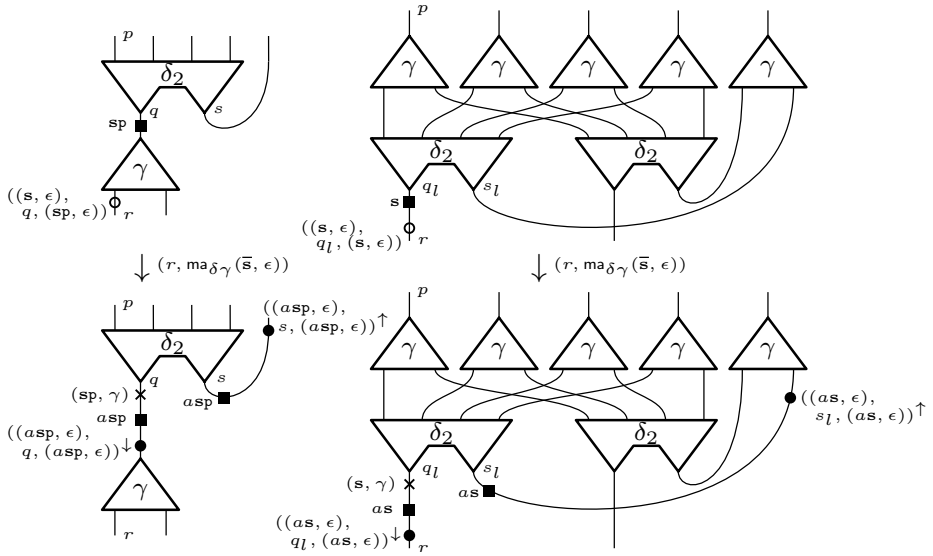
- A marriage between a principal port and a free port provokes one marriage action, whose marriage type is either $\gamma\delta$, $\delta\delta$ or $\delta\gamma$. The pictures below illustrate how π simulates the external marriage in ρ for each marriage type.
  - $\gamma\delta$



  - $\delta\delta$



  - $\delta\gamma$

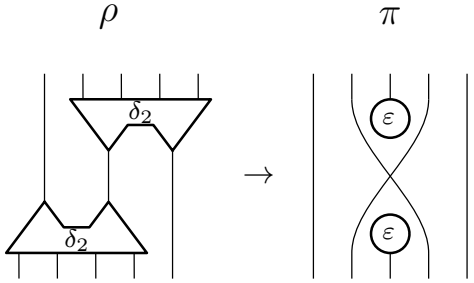


- For the other direction, we can proceed similarly.

□

Lemma 8. *If $\rho \to_{\delta\delta} \pi$, then there is t_ρ such that $s_\rho \to t_\rho$ by an internal marriage transition and $(\mathbb{FM}(\mathcal{TKS}^\rho), \to, t_\rho) \approx \mathsf{TM}_\pi$.*

Proof. Assume that the reduction is an interaction between two first principal ports of two δ_2 cells.



We can see s_ρ immediately evolves into t_ρ, which consists of 2 married status tokens and 2 single status tokens. We define a relation $\mathcal{R} \subseteq \mathbb{FM}(\mathcal{TKS}^\rho) \times \mathbb{FM}(\mathcal{TKS}^\pi)$ as:

- Matching tokens are related naturally.
- Marriage tokens in ρ which originated in free ports and are t_ρ-canonical are related to the natural counterparts in π for their origins.

The bisimulation between t_ρ and s_π is given by $\mathcal{R}' = \{(\mathsf{S}_\rho \uplus t_\rho, \mathsf{S}_\pi) \mid (\mathsf{S}_\rho, \mathsf{S}_\pi) \in \mathcal{R}\}$. The proof that $\mathcal{R}'$ is a bisimulation is very much similar to the one of Lemma 6. □

One may wonder why a $\gamma\delta$ reduction is treated the same as a $\delta\delta$ reduction, since no nondeterminism seems to be involved in it. This is however, only apparent: if a δ_2 cell may "choose" to interact with either a δ_2 or a γ cell, it does not loose any behaviour *only up to η-equivalence* (see [33] for a definition) when choosing to interact with the γ cell. GoI, on the other hand, is well-known *not* to be sound for η in general. In other words, although $\gamma\delta$ reduction is *deterministic* as far as rewriting on nets is concerned, it is not so at the level of the underlying GoI model, and this happens for deep reasons which have little to do with MICs.

As a corollary of the three lemmas above, we can prove soundness of our model with respect to net reduction, which is spelled out as a similarity:

Corollary 1 (Soundness)**.** *If $\rho \to \pi$, then $\mathsf{TM}_\pi \lessapprox \mathsf{TM}_\rho$.*

Proof. This follows from Lemma 6, Lemma 7, and Lemma 8. Indeed, if $s_\rho \to t_\rho$, then $(\mathbb{FM}(\mathcal{TKS}^\rho), \to, t_\rho) \lessapprox \mathsf{TM}_\rho$. □

6 Adequacy

Corollary 1 tells us that, along any interaction path (of which there could be many) starting from any net ρ, one finds nets whose token machines can all be related by way of weak similarity to TM_ρ. This already tells us much about the way nets and token machines are related. However, there is still no result around about the relationship between the behaviour of TM_ρ and that of ρ itself, e.g., any result on whether there is a way to "read" a property of ρ from its interpretation TM_ρ. This section is devoted to proving that, indeed, ρ and its token machine have the same behaviour as for *termination*. This means that token machines are not only a sound but also an *adequate* model of multiport interaction, since termination is the most natural property to be observed.

Before stating adequacy, we need to introduce some more preliminary definitions and notations. In particular, in this section, contrarily to the previous ones, token machines will be seen as non-interactive objects, and thus analysed as *closed* systems. A *(unlabelled) transition system* (TS for short; also known as a *Kripke structure*) consists of a set S of *states* and a binary relation $\rightarrow \subseteq S \times S$. A *pointed* transition system is a triple $(S, \rightarrow, s)$ where $(S, \rightarrow)$ is a TS and $s \in S$ is the *initial state*. Notions of simulation and bisimulation can easily be given for pointed transition systems. Noticeably, in a simulation we require each transition in the simulated TS to correspond to *zero or more* transitions in the simulating TS. We indicate the obtained notions of (weak) bisimilarity and similarity as $\approx$ and $\lessapprox$, as usual. Given a transition system $\mathbf{A} = (S, \rightarrow)$, we define the following two predicates parametrised by an element S of S and by a set of elements $\mathbf{X}$:

- $fin^{\mathbf{X}}_{\mathbf{A}}(\mathsf{S})$, which holds iff *there is* a *finite* transition sequence starting in S and ending in a normal form *not in* $\mathbf{X}$.
- $inf^{\mathbf{X}}_{\mathbf{A}}(\mathsf{S})$, which instead holds iff *there is* an *infinite* transition sequence starting in S, or a finite transition sequence starting in S and ending in an element *in* $\mathbf{X}$.

NETS can be seen as a transition system whose states are nets, and whose transition relation is the one induced by the reduction relation. Let $\mathbf{VC}$ be the set of nets containing vicious circles. For any net ρ, the set TM_ρ can itself be seen as a transition system whose states are elements of $\mathcal{TKS}^\rho$ and whose transition relation consists of internal transitions.

Theorem 2 (Adequacy)**.** *For every net ρ, the following holds:*

$$fin^{\emptyset}_{\mathsf{TM}_\rho}(s_\rho) \Longleftrightarrow fin^{\mathbf{VC}}_{\mathsf{NETS}}(\rho);$$
$$inf^{\emptyset}_{\mathsf{TM}_\rho}(s_\rho) \Longleftrightarrow inf^{\mathbf{VC}}_{\mathsf{NETS}}(\rho).$$

The rest of this section will be devoted to proving the four implications which together form Theorem 2.

6.1 Standard Computations

In this section we define a key notion, called *standard computation*, which plays a crucial role to prove Adequacy. In the following, a *computation* means a transition sequence starting in the initial state of the underlying token machine.

A *thunk* is any transition sequence in one of two forms:

- Either in the form Θ, Ξ, Π, where

$$\Theta : T_1 \underset{\mathsf{S}}{\rightarrow} T_2 \underset{\mathsf{S}}{\rightarrow} \cdots \underset{\mathsf{S}}{\rightarrow} T_n;$$
$$\Xi : X_1 \underset{\mathsf{U}}{\rightarrow} X_2 \underset{\mathsf{U}}{\rightarrow} \cdots \underset{\mathsf{U}}{\rightarrow} X_m$$

  where T_1 and X_1 are origin tokens, and Π is a $\delta\delta$ marriage whose main tokens are T_n and X_m;
- Or in the form Θ, Ξ, where

$$\Theta : T_1 \underset{\mathsf{S}}{\rightarrow} T_2 \underset{\mathsf{S}}{\rightarrow} \cdots \underset{\mathsf{S}}{\rightarrow} T_n$$

  T_1 is an origin token, and Ξ is a $\gamma\delta$ marriage whose main token is T_n.

A *standard sequence* is a transition sequence in one of two forms:

- Either in the form $\Theta_1, \ldots, \Theta_n, \Xi$ where $\Theta_1, \ldots, \Theta_n$ are all thunks, and Ξ is any *maximal* transition sequence *not* containing any marriage step, called a *continuation*.
- Or in the form $\Theta_1, \Theta_2, \ldots$, where all the Θ_i (where $i \in \mathbb{N}^+$) is a thunk.

A *standard computation* is a computation which is a standard sequence. The key insight is that any maximal computation can be "represented" by a standard computation. Formally, we have the following lemma:

Lemma 9 (Standardisation)**.** • *Let* $\Theta\colon s_{\mathsf{TM}_\rho} \to \mathsf{S}_1 \to \mathsf{S}_2 \to \cdots \to \mathsf{S}_n$ *be a finite maximal computation of* TM_ρ*. There exists a finite standard computation* $\Xi\colon s_{\mathsf{TM}_\rho} \to \mathsf{U}_1 \to \mathsf{U}_2 \to \cdots \to \mathsf{U}_m$ *s.t.* $\mathsf{U}_m = \mathsf{S}_n$*.*

• *Let* $\Theta : s_{\mathsf{TM}_\rho} \to \mathsf{S}_1 \to \mathsf{S}_2 \to \cdots$ *be an infinite maximal computation of* TM_ρ*. There exists an infinite standard computation* $\Xi\colon s_{\mathsf{TM}_\rho} \to \mathsf{U}_1 \to \mathsf{U}_2 \to \cdots$ *s.t. for any state* S_i *right after a marriage in* Θ*, there exists a state* $\mathsf{U}_{i'} = \mathsf{S}_i$ *in* Ξ*.*

Proof. First we show the following auxiliary propositions.

Proposition 3. *In any token machine, the following three hold.*

1. *Let* $\mathsf{S}_0 \to \mathsf{S}_1$ *be a move on a token* T *and* $\mathsf{S}_1 \to \mathsf{S}_2$ *be a move on another* $X \neq T$*. Then there exists a state* S'_1 *satisfying* $\mathsf{S}_0 \to \mathsf{S}'_1 \to \mathsf{S}_2$*, where* $\mathsf{S}_0 \to \mathsf{S}'_1$ *is a move on* X *and* $\mathsf{S}'_1 \to \mathsf{S}_2$ *is a move on* T*.*
2. *Let* $\mathsf{S}_1 \to \mathsf{S}_2$ *be a marriage and* $\mathsf{S}_0 \to \mathsf{S}_1$ *be a move on a token* T *not involved in the marriage* $\mathsf{S}_1 \to \mathsf{S}_2$*. Then there exists a state* S'_1 *satisfying* $\mathsf{S}_0 \to \mathsf{S}'_1 \to \mathsf{S}_2$*, where* $\mathsf{S}_0 \to \mathsf{S}'_1$ *is a marriage on the same token(s) as the one* $\mathsf{S}_1 \to \mathsf{S}_2$*, and* $\mathsf{S}'_1 \to \mathsf{S}_2$ *is a move on* T*.*
3. *Let* $\mathsf{S}_1 \to \mathsf{S}_2$ *be a marriage and* $\mathsf{S}_0 \to \mathsf{S}_1$ *be a move on a token* T *killed by the marriage* $\mathsf{S}_1 \to \mathsf{S}_2$*. Then* $\mathsf{S}_0 \to \mathsf{S}_2$*, by a marriage on the same token(s).*

Proof.
1. Any move transition does not affect the other token's move transition. This is in particular because any move transition does not put any marriage status token by definition.
2. Since $\mathsf{S}_0 \to \mathsf{S}_1$ is already allowed to happen before the marriage $\mathsf{S}_1 \to \mathsf{S}_2$, there is already some marriage status token(s) that allow T to move in S_0. Since any marriage does not delete a marriage status token by definition, the move is still possible in S'_1.
3. Since $\mathsf{S}_0 \to \mathsf{S}_1$ is a move, it does not put any new token nor change any other token's stacks by definition.

□

Let $\Theta\colon \mathsf{S}_0 \to \mathsf{S}_1 \to \cdots \to \mathsf{S}_k \to \mathsf{S}_{k+1} \to \mathsf{S}_{k+2} \to \cdots \to \mathsf{S}_n$ be a finite maximal transition sequence where $\mathsf{S}_k \to \mathsf{S}_{k+1}$ is the first marriage in the sequence Θ. We first permute Θ, by repeatedly applying Proposition 3.1, into another transition sequence $\Theta'\colon \mathsf{S}_0 \to^* \mathsf{S}'_1 \to^* \mathsf{S}'_2 \to^* \mathsf{S}_k \to \mathsf{S}_{k+1} \to \mathsf{S}_{k+2} \to \cdots \to \mathsf{S}_n$, where $\mathsf{S}_0 \to^* \mathsf{S}'_1$ is one (if the marriage is $\gamma\delta$) or two (if the marriage is $\delta\delta$) consecutive focused sequences on the token(s) involved in the marriage, $\mathsf{S}'_1 \to^* \mathsf{S}'_2$ consists only of moves on tokens not involved in the marriage, $\mathsf{S}'_2 \to^* \mathsf{S}_k$ consists only of moves on tokens killed by the marriage, and the rest is the same as Θ. Then, applying Proposition 3.2 and 3.3, we obtain a transition sequence $\Theta''\colon \mathsf{S}_0 \to^* \mathsf{S}'_1 \to \mathsf{S}'_2 \to^* \mathsf{S}_{k+1} \to \mathsf{S}_{k+2} \to \cdots \to \mathsf{S}_n$, where $\mathsf{S}_0 \to^* \mathsf{S}'_1$ is focused sequences on the token(s) involved in the marriage $\mathsf{S}'_1 \to \mathsf{S}'_2$, i.e. with a thunk at its head. If the subsequence $\mathsf{S}'_2 \to^* \mathsf{S}_{k+1} \to \mathsf{S}_{k+2} \to \cdots \to \mathsf{S}_n$ still contains marriages, apply the procedure described above inductively.

□

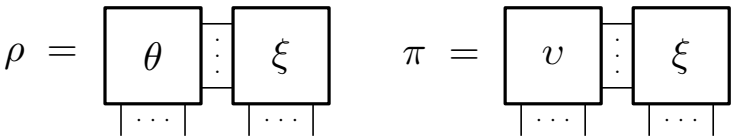
6.2 Preservation Properties

Given $\rho \to \pi$, one can define a relation $\mathcal{R}^\rho_\pi$ and prove it having certain bisimulation properties, as described in Section 5. The relations are actually useful here, too; in this section, we show various properties on a reduction and transition sequences using the relations $\mathcal{R}^\rho_\pi$. By definition, the relations $\mathcal{R}^\rho_\pi$ are all defined as set of pairs $(\mathsf{S}_\rho, \mathsf{S}_\pi)$, where marriage tokens in S_ρ have a natural counterpart in S_π, and *vice versa*. The fact that $\mathcal{R}^\rho_\pi$ are weak bisimulation relations implies that if $(\mathsf{S}_\rho, \mathsf{S}_\pi) \in \mathcal{R}^\rho_\pi$ and Θ is a transition sequence starting in S_ρ and ending in U_ρ, then there are some naturally defined sequences starting in S_π and ending in U_π such that $(\mathsf{U}_\rho, \mathsf{U}_\pi) \in \mathcal{R}^\rho_\pi$. In this case we say that $\mathcal{R}^\rho_\pi$ *sends* Θ to any of those transition sequences or, simply, that Θ *is sent* to any of them. Conversely if Ξ is a transition sequence starting in S_π and ending in U_π, then there are some naturally defined sequences starting from S_ρ and ending in U_ρ such that $(\mathsf{U}_\rho, \mathsf{U}_\pi) \in \mathcal{R}^\rho_\pi$. In this case we say that $\mathcal{R}^\rho_\pi$ *sends back* any of these sequences to Θ or, simply, that any of these sequences *is*

sent back to Θ. All this can be easily generalised to infinite transition sequences. If a transition sequence Θ in a collection *RED* is sent (sent back, respectively) to a transition Ξ, not much can be said, in general, about whether $\Xi \in RED$. If, however, *for every* $\Theta \in RED$, *there is* $\Xi \in RED$ such that Θ is sent (sent back, respectively) to Ξ, we say that *RED* (or the predicate on transition sequences defining *RED*) is *preserved forward* (*preserved backward*, respectively).

Lemma 10. *Infinite focused sequences starting in reachable states are preserved forward and backward.*

Proof. Suppose that $\rho \to \pi$. Then, ρ can be seen as a redex θ in parallel to the rest of the net ξ, while π can be seen as a reduct υ in parallel to ξ, graphically:



The redex θ can be any pair of cells facing each other, together with wires linking some of the auxiliary ports of the two cells, if this is the way they are linked in ρ. Now, consider an infinite focused sequence Θ in ρ. We can have two cases:

- Θ goes back and forth between θ and ξ *infinitely* often. In this case, it is easy to realise that Θ is sent to an infinite focused sequence in π. Indeed, each of the infinitely many finite portions of it in θ (respectively, ξ) is mapped to a finite corresponding portion in υ (respectively, ξ). Moreover, each of the finite portions of Θ in ξ is non-empty (because if it is empty, the corresponding wire would be part of θ) and is sent to itself.
- Θ goes back and forth between θ and ξ only *finitely* often. In this case, there needs to be an infinite suffix Ξ of Θ which lies in one of the two components, which however must be ξ, because no infinite focused sequence starting in a reachable state can be constructed for a redex. As a consequence, Θ is sent forward to an infinite focused sequence in π, because Ξ is also a transition sequence of π.

A similar argument can be given for every infinite focused sequence Θ in π. □

Lemma 11. *Thunks are preserved forward and backward.*

Proof. The first thing to observe here is that the left projection of $\mathcal{R}^{\rho}_{\pi}$ only includes states in which the marriage corresponding to the redex leading ρ to π has already been performed. Once we realise this, showing that thunks are preserved forward and backward amounts to the following observations:

- First of all, $\mathcal{R}^{\rho}_{\pi}$ puts origin tokens in correspondence with origin tokens, and states for which marriages are ready to states with the same property.
- Second of all, focused sequences are sent (and sent back) to focused sequences.

This concludes the proof. □

Lemma 12. *Continuations starting in a reachable state are preserved backward and forward.*

Lemma 13. *Infinite continuations starting in a reachable state are preserved backward and forward.*

Proof. We can first of all observe that continuations are preserved backward and forward. Now, suppose by way of contradiction, that an *infinite* continuation Θ is sent to a *finite* continuation Ξ. Since in any continuation, marriages are not allowed to happen, it means that Θ (respectively, Ξ) can be seen as the interleaving of a finite number of focused reductions $\Theta_1, \ldots, \Theta_n$ (respectively, $\Xi_1, \ldots, \Xi_n$) such that Θ_i is mapped to Ξ_i. Now, at least one of the Θ_i, say Θ_j must be infinite, otherwise Θ would be finite itself. By Lemma 10, Ξ_j is thus infinite, and as a consequence the whole of Ξ is infinite. □

Proposition 4. *Suppose that* $\rho \to_a \pi$ *where* $a \in \{\delta\delta, \gamma\delta\}$. *If* TM_{π} *has a finite standard computation with* n *thunks then* TM_{ρ} *has a finite standard computation with* $n + 1$ *thunks.*

Proof. By Lemma 7 and Lemma 8, there is t_ρ such that $s_\rho \to t_\rho$ and $(t_\rho, s_\pi) \in \mathcal{R}^{\rho}_{\pi}$. Since thunks and continuations are preserved backward, the thesis easily follows. □

Proposition 5. *Suppose that $\rho \to_{\gamma\gamma} \pi$. If TM_π has a standard computation with n thunks, then TM_ρ also has one.*

Proof. By Lemma 6, it holds that $\mathsf{TM}_\rho \approx \mathsf{TM}_\pi$. Since thunks and continuations are preserved backward, the thesis easily follows. □

Proposition 6. *If TM_ρ has a finite standard computation which is a continuation, then $\rho \to^* \pi$, where π is in normal form and $\pi \notin \mathbf{VC}$.*

Proof. Consider the net π obtained by reducing all $\gamma\gamma$ redexes in ρ, in an arbitrary order. This process of course terminates, because each reduction step makes the number of wires in the net to decrease. Now:

- π is in normal form. If it contains a $\delta\delta$ or $\delta\gamma$ redex, indeed, TM_π would admit a standard computation with at least one thunk. Since thunks are preserved backward, one would get the same for TM_ρ. This is however incompatible with TM_ρ having a finite standard computation which is a continuation.
- π does not contain any γ-vicious circle. If it contains one, indeed, TM_π would admit an *infinite* standard computation which is a continuation, and then also TM_ρ would admit one. This is again incompatible with TM_ρ having a finite standard computation which is a continuation.

This concludes the proof. □

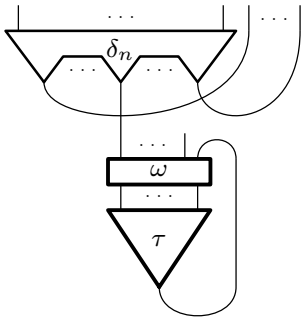
Proposition 7. *If TM_ρ has a standard computation with $n+1$ thunks and a continuation Θ (respectively, with infinitely many thunks), then $\rho \to^+ \pi$, where TM_π has a standard computation with n thunks and a continuation Ξ (respectively, with infinitely many thunks). Moreover Θ is infinite iff Ξ is.*

Proof. First of all, let us consider the net θ one obtains by reducing all $\gamma\gamma$ redexes in ρ, in an arbitrary order. This process of course terminates, because each reduction step makes the number of wires in the net to decrease. The net θ has itself a standard computation with $n+1$ thunks and a continuation (respectively, it has infinitely many thunks). Since θ only contains $\delta\delta$ or $\delta\gamma$ redexes, the first thunk in the standard computation corresponds to one of redex of one of those kinds. As a consequence there is π such that $\theta \to \pi$. Moreover, by Lemma 7 and Lemma 8, π admits a standard computation Ξ with n thunks and a continuation (respectively, with infinitely many thunks), which is infinite iff Θ is. □

Proposition 8. *Let ρ be a net in normal form. If $\rho \notin \mathbf{VC}$, then TM_ρ only admits finite computations. Otherwise, all maximal computation of TM_ρ are infinite.*

Proof. Assume $\rho \notin \mathbf{VC}$. In this situation, a marriage token cannot hit any principal port, since it means there remains a redex (and thus there will be no marriage). Thus a marriage token can only hit auxiliary ports one by one; if it hits an auxiliary port of a δ_2 cell, it stops. Moreover there is no way to make it able to move on, since no marriage can take place. Therefore the only possibility for a token to travel infinitely is to go through γ cells infinitely many times, each time from an auxiliary port to the principal port. However, since ρ is finite, this means the token is trapped into a loop consisting of γ cells, which means there is a γ-vicious circle, which in turn contradicts to $\rho \notin \mathbf{VC}$. Hence a marriage token can perform transitions only finitely many times. Considering that the number of marriage tokens is also finite and it does not increase, all computations of TM_ρ are finite.

Assume $\rho \in \mathbf{VC}$. Recall it means ρ contains a subnet in the following form:



Since ν contains one or more δ_2 cells, they emit marriage tokens and at least one of them reaches σ since ν is a tree. It reaches one of the leaves of τ since σ is just a wiring. Finally, the token necessarily reaches the root of τ and is feeded back to σ, since τ consists only of γ cells; this continues forever. By the same reason as the case of $\rho \notin \mathbf{VC}$, there will not be any marriage, hence the infinite trip described above cannot end by being killed. □

Proposition 9. *If TM_ρ has an infinite standard computation which is a continuation, then there exists π satisfying $\rho \to^* \pi$ solely by $\gamma\gamma$ reductions and $\pi \in \mathbf{VC}$.*

Proof. First, we reduce $\gamma\gamma$ redexes as much as possible, obtaining $\rho \to^* \pi$. By Lem. 13, TM_π also has an infinite standard computation which is a continuation, say Θ. Since Θ is a continuation, by definition it does not contain any marriage transition. Thus the number of marriage tokens is finitely bounded, implying there exists at least one token that moves infinitely many times. Let us follow the token's infinite path. The token starts from a principal port of a δ_2 cell with an empty configuration. then it can*not* hit

- a principal port (of any kind of cells) since then a marriage happens.
- an auxiliary port of a δ_2 cell, since Θ does not contain a marriage and thus it stops.

So it must hit an auxiliary port of a γ cell and goes out from its principal port, now with a non-empty configuration. After passing a γ cell, it can*not* hit

- a principal port of a δ_2 cell or an ε cell since it stops due to γ stack being not empty.
- a principal port of γ cell since then π has a $\gamma\gamma$ redex.
- an auxiliary port of a δ_2 cell, because Θ does not contain a marriage and thus it stops.

So again it must hit an auxiliary port of a γ cell and goes out from its principal port with a non-empty configuration. Since π is finite, the net π necessarily has a loop consisting only of γ cells connected with a tree containing a δ_2 cell, i.e. a γ-vicious circle. □

6.3 The Four Implications

Lemma 14. $\mathit{inf}^{\mathbf{VC}}_{\mathsf{NETS}}(\rho) \Longrightarrow \mathit{inf}^{\emptyset}_{\mathsf{TM}_\rho}(s_{\mathsf{TM}_\rho})$

Proof. If $\mathit{inf}^{\mathbf{VC}}_{\mathsf{NETS}}(\rho)$, then we can distinguish two cases:

- Either ρ admits an infinite reduction sequence. In this case, we prove that $\mathit{inf}^{\emptyset}_{\mathsf{TM}_\rho}(s_{\mathsf{TM}_\rho})$ by coinduction. Indeed, if ρ admits an infinite reduction sequence, then there is a finite reduction

$$\rho \to^*_{\gamma\gamma} \pi \to_a \theta$$

  where $a \in \{\delta\gamma, \delta\delta\}$ and $\mathit{inf}^{\mathbf{VC}}_{\mathsf{NETS}}(\theta)$. We know, by soundness, that $\mathsf{TM}_\rho \approx \mathsf{TM}_\pi$ and that s_π reduces in *at least* one step to a state t which is bisimilar to s_θ. We can then conclude by coinduction that since $\mathit{inf}^{\emptyset}_{\mathsf{TM}_\theta}(s_{\mathsf{TM}_\theta})$, it holds that $\mathit{inf}^{\emptyset}_{\mathsf{TM}_\rho}(s_{\mathsf{TM}_\rho})$.
- Or ρ admits a finite reduction sequence to a net π in $\mathbf{VC}$. The net π contains a γ-vicious circle, and then $\mathit{inf}^{\emptyset}_{\mathsf{TM}_\pi}(s_{\mathsf{TM}_\pi})$ holds. By induction on the number of reduction steps leading ρ to π, one can then prove that $\mathit{inf}^{\emptyset}_{\mathsf{TM}_\rho}(s_{\mathsf{TM}_\rho})$.

This concludes the proof. □

Lemma 15. $\mathit{fin}^{\mathbf{VC}}_{\mathsf{NETS}}(\rho) \Longrightarrow \mathit{fin}^{\emptyset}_{\mathsf{TM}_\rho}(s_{\mathsf{TM}_\rho})$

Proof. If $\mathit{fin}^{\mathbf{VC}}_{\mathsf{NETS}}(\rho)$, then ρ reduces in n steps to a normal net not in $\mathbf{VC}$. One can prove, that, for any such net ρ, TM_ρ admits a finite computation:

- First of all, if ρ is itself a normal form not in $\mathbf{VC}$, then ρ admits a finite (standard) computation, due to Proposition 8.
- If $\rho \to \pi$ and TM_π admits a finite standard computation, then TM_ρ itself admits a finite standard computation, due to Proposition 4 and Proposition 5.

□

Lemma 16. $\mathit{inf}^{\emptyset}_{\mathsf{TM}_\rho}(s_{\mathsf{TM}_\rho}) \Longrightarrow \mathit{inf}^{\mathbf{VC}}_{\mathsf{NETS}}(\rho)$

Proof. If there is an infinite computation from s_{TM_ρ}, then:

- Either there is an infinite standard computation with a finite number n of thunks and an infinite continuation from s_{TM_ρ}. Then, by induction on n, one can prove that ρ reduces to another net π such that TM_π has an infinite standard reduction which is a continuation. By Proposition 9, this implies the thesis.
- Or there is an infinite standard computation with infinitely many thunks from s_{TM_ρ}. Then, we can apply Proposition 7 in a coinductive way to get an infinite reduction from ρ.

□

Lemma 17. $fin^{\emptyset}_{\mathsf{TM}_\rho}(s_{\mathsf{TM}_\rho}) \Longrightarrow fin^{\mathbf{VC}}_{\mathsf{NETS}}(\rho)$

Proof. If there is a finite computation from s_{TM_ρ}, then there is also a finite standard computation with n thunks from s_{TM_ρ}. We can then proceed by induction on n, thanks to Proposition 6 and Proposition 7. If $n = 0$ then ρ is in normal form not in **VC** by Proposition 6; if $n > 0$ then there exists π satisfying $\rho \rightarrow^* \pi$ and TM_π admits a finite standard computation with $n - 1$ thunks by Proposition 7. □

7 Discussion

A formal comparison between multi-token machines and other concurrent models of computation, and in particular the so-called truly concurrent ones, is outside the scope of this paper. Some observations in this direction are anyway useful, and are the starting point of current investigations by the authors.

Multi-token machines as we defined them in this paper can be seen as Petri nets where, however, the set of *places* and *transitions* are both denumerable. In particular, places correspond to the possible states of a token, of which there can be countably many: remember that marriage tokens carry two stacks of unbounded length with them. Indeed, tokens lying at the same wire, but with different stacks need to correspond to different places at the level of Petri nets, because they can travel in completely different directions. Implementing the marriage mechanism in Petri nets is indeed possible, but requires *nonlocal* transitions, transitions with incoming arcs coming from places which are "far away" from each other: actually, there must be one such transition for any pair of (virtual copies of) δ_2 principal ports in the underlying net. In other words, the nonlocal behaviour of certain multi-token machine interactions would be reflected, at the level of Petri nets, by a possibly very complex and nonlocal system of transitions. Apart from that, the encoding seems possible, and indeed targets plain Petri nets rather than the kind of Petri nets that are usually required to model process algebras like the π-calculus, e.g., Petri nets with inhibitor arcs [7]. Whether a finitary π-calculus can be given a plain Petri net semantics this way is an interesting open problem.

Perhaps even more direct is the correspondence between multi-token machines and chemical abstract machines as defined by, e.g., Berry and Boudol [6]. In this view, tokens would become *molecules*, and would be allowed to float in a *soup*, freely interacting between them. Marriage transitions of multi-token machines would be modelled by chemical rules allowing two matching molecules (i.e. marriage tokens) to interact, producing some other tokens. Moving transitions, on the other hand, are most often possible in presence of a status token, and this mechanism itself can be seen as a chemical reaction (which leave one of the two involved ingredients essentially unchanged). In this view, the underlying net's wires would become molecule kinds, and the obtained abstract machine would be syntax-free.

8 Conclusions

In this paper, we introduced the first truly concurrent geometry of interaction model, by defining multi-token machines for Mazza's multiport interaction combinators, and by proving them to be sound and adequate. Mazza himself suggested this as an open problem [33]. What is interesting about our results is also the way they are spelled out and proved, namely by way of labelled transition systems and (bi)similarity. This is something quite natural, but new to geometry of interaction models, at least to the authors' knowledge.

As already mentioned, one topic for future work is a study of multi-token machines for differential interaction nets, and in particular to a "multiport" variation of them, something the authors are actually working on as a way towards a better modelling of the π-calculus. Another topic of future work concerns

the study of the concurrent nature of multi-token machines, and in particular of their causal structure, especially in view of their interpretation as Petri nets.

Acknowledgment

The first author is partially supported by the ANR projects 12IS02001 PACE, and 14CE250005 ELICA. The third author is partially supported by the JST ERATO Grant Number JPMJER1603. The three authors are supported by the INRIA-JSPS joint project CRECOGI.

References

[1] Samson Abramsky, Kohei Honda, and Guy McCusker. A fully abstract game semantics for general references. In *Proc. of LICS 1998*, pages 334–344. IEEE Computer Society, 1998.

[2] Samson Abramsky and Radha Jagadeesan. A game semantics for generic polymorphism. In *Proc. of FOSSACS 2003*, volume 2620 of *LNCS*, pages 1–22. Springer, 2003.

[3] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Inf. Comput.*, 163(2):409–470, 2000.

[4] Samson Abramsky and Paul-André Melliès. Concurrent games and full completeness. In *Proc. of LICS 1999*, pages 431–442. IEEE Computer Society, 1999.

[5] Vladimir Alexiev. *Non-deterministic interaction nets*. PhD thesis, University of Alberta, 1999.

[6] Gérard Berry and Gérard Boudol. The chemical abstract machine. *Theor. Comput. Sci.*, 96(1):217–248, 1992.

[7] Nadia Busi and Roberto Gorrieri. Distributed semantics for the pi-calculus based on petri nets with inhibitor arcs. *J. Log. Algebr. Program.*, 78(3):138–162, 2009.

[8] Pierre Clairambault, Julian Gutierrez, and Glynn Winskel. The winning ways of concurrent games. In *Proc. of LICS 2012*, pages 235–244. IEEE Computer Society, 2012.

[9] Ugo Dal Lago, Claudia Faggian, Ichiro Hasuo, and Akira Yoshimizu. The geometry of synchronization. In *Prof. of CSL-LICS 2014*, pages 35:1–35:10. ACM, 2014.

[10] Ugo Dal Lago, Claudia Faggian, Benoît Valiron, and Akira Yoshimizu. Parallelism and synchronization in an infinitary context. In *Prof. of LICS 2015*, pages 559–572. IEEE Computer Society, 2015.

[11] Ugo Dal Lago, Claudia Faggian, Benoît Valiron, and Akira Yoshimizu. The geometry of parallelism. classical, probabilistic, and quantum effects. In *Proc. of POPL 2017*, pages 833–845. ACM, 2017.

[12] Ugo Dal Lago, Ryo Tanaka, and Akira Yoshimizu. The geometry of concurrent interaction (extended version). Available at `http://eternal.cs.unibo.it/gci.pdf`, 2016.

[13] Vincent Danos and Laurent Regnier. Reversible, irreversible and optimal lambda-machines. *Theor. Comput. Sci.*, 227(1-2):79–97, 1999.

[14] Marc de Falco. The geometry of interaction of differential interaction nets. In *Proc. of LICS 2008*, pages 465–475. IEEE Computer Society, 2008.

[15] Andrei Dorman and Damiano Mazza. A hierarchy of expressiveness in concurrent interaction nets. In *Proc. of CONCUR 2013*, volume 8052 of *LNCS*, pages 197–211. Springer, 2013.

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

[17] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. *Inf. Comput.*, 208(6):606–633, 2010.

[18] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Electr. Notes Theor. Comput. Sci.*, 123:35–74, 2005.

[19] Olle Fredriksson and Dan R. Ghica. Abstract machines for game semantics, revisited. In *Proc. of LICS 2013*, pages 560–569. IEEE Computer Society, 2013.

[20] Dan R. Ghica. Geometry of synthesis: a structured approach to VLSI design. In *Proc. of POPL 2007*, pages 363–375. ACM, 2007.

[21] Dan R. Ghica and Andrzej S. Murawski. Angelic semantics of fine-grained concurrency. In *Prof. of FOSSACS 2004*, volume 2987 of *LNCS*, pages 211–225. Springer, 2004.

[22] Dan R. Ghica and Alex I. Smith. Geometry of synthesis II: from games to delay-insensitive circuits. *Electr. Notes Theor. Comput. Sci.*, 265:301–324, 2010.

[23] Jean-Yves Girard. Geometry of interaction 1: Interpretation of system F. In S. Valentini R. Ferro, C. Bonotto and A. Zanardo, editors, *Proc. of Logic Colloquium 1988*, volume 127 of *Studies in Logic and the Foundations of Mathematics*, pages 221 – 260. Elsevier, 1989.

[24] Georges Gonthier, Martín Abadi, and Jean-Jacques Lévy. Linear logic without boxes. In *Proc. of LICS 1992*, pages 223–234. IEEE Computer Society, 1992.

[25] Naohiko Hoshino, Koko Muroya, and Ichiro Hasuo. Memoryful geometry of interaction: from coalgebraic components to algebraic effects. In *Prof. of CSL-LICS 2014*, pages 52:1–52:10. ACM, 2014.

[26] J. M. E. Hyland and C.-H. Luke Ong. On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.

[27] Yves Lafont. Interaction combinators. *Inf. Comput.*, 137(1):69–101, 1997.

[28] James Laird. A game semantics of idealized CSP. *Electr. Notes Theor. Comput. Sci.*, 45:232–257, 2001.

[29] John Lamping. An algorithm for optimal lambda calculus reduction. In *Proc. of POPL 1990*, pages 16–30. ACM Press, 1990.

[30] Ian Mackie. The geometry of interaction machine. In *Proc. of POPL 1995*, pages 198–208. ACM Press, 1995.

[31] Damiano Mazza. The true concurrency of differential interaction nets. *Mathematical Structures in Computer Science*. To Appear.

[32] Damiano Mazza. Multiport interaction nets and concurrency. In *Proc. of CONCUR 2005*, volume 3653 of *LNCS*, pages 21–35. Springer, 2005.

[33] Damiano Mazza. *Interaction Nets: Semantics and Concurrent Extensions*. PhD thesis, Université de la Méditerranée and Università degli Studi Roma Tre, 2006.

[34] Paul-André Melliès. Asynchronous games 2: The true concurrency of innocence. In *Proc. of CONCUR 2004*, volume 3170 of *LNCS*, pages 448–465. Springer, 2004.

[35] Koko Muroya, Naohiko Hoshino, and Ichiro Hasuo. Memoryful geometry of interaction II: recursion and adequacy. In *Proc. of POPL 2016*, pages 748–760. ACM, 2016.

[36] Carl Adam Petri. Communication with automata. Technical Report RADC-TR-65-377, Rome Air Dev. Center, New York, 1966. English translation of the original PhD thesis (1962).

[37] Silvain Rideau and Glynn Winskel. Concurrent strategies. In *Proc. of LICS 2011*, pages 409–418. IEEE Computer Society, 2011.

[38] Davide Sangiorgi. From pi-calculus to higher-order pi-calculus - and back. In *Proceedings of TAPSOFT 1993*, volume 668 of *LNCS*, pages 151–166. Springer, 1993.

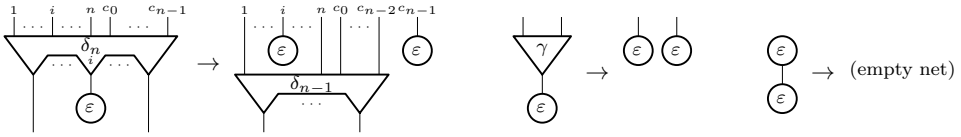


Figure 14: MIC Interaction Rules with ε

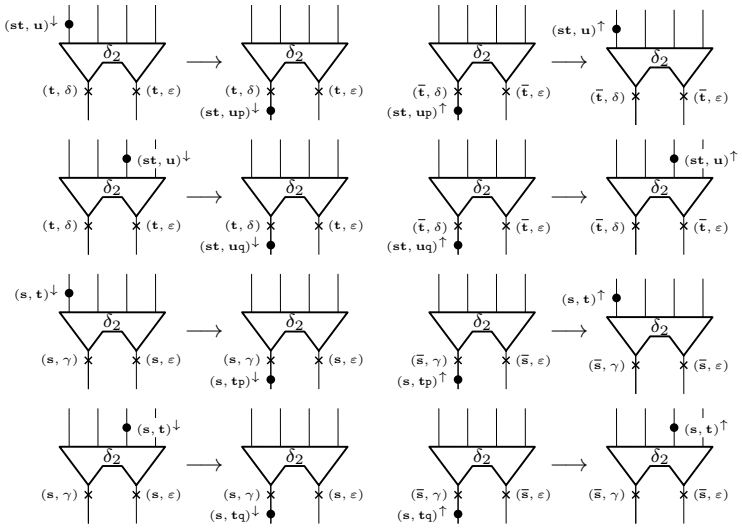


Figure 15: Internal Transition Rules with ε

Appendix

In this section, we will give some more details as for why all the results we proved in this paper can be generalised to MIC where ε cells are allowed to interact with other cells (including themselves).

Interaction Rules. First of all, we need some additional interaction rules, namely those in Figure 14. Please notice that this new sets of rules taken in isolation turns MICs into a confluent and strongly normalizing rewrite system. Actually, the first of the three new rules implies that we cannot work with δ_2 cells only, but also with δ_1 cells.

Token Machines. As for token machines, we need to substantially generalise the ones we introduced in the paper. First of all, cell types need to be generalised. Moreover, more reduction rules are needed, because δ cells can now marry not only with δ or γ cells, but also with ε cells. They are in Figure 15, Figure 16, and Figure 17. The proof of the compositionality remains essentially unaltered, except for the fact that more cases need to be taken into account.

Soundness. The proof of soundness needs to be adapted. In particular, we need to prove lemmas analogous to Lemma 6, Lemma 7 and Lemma 8, but for interaction rules involving ε cells. Here they are:

Lemma 18. *If $\rho \to_{\gamma\varepsilon} \pi$, then $\mathsf{TM}_\rho \approx \mathsf{TM}_\pi$.*

Lemma 19. *If $\rho \to_{\varepsilon\varepsilon} \pi$, then $\mathsf{TM}_\rho \approx \mathsf{TM}_\pi$.*

Lemma 20. *If $\rho \to_{\delta\varepsilon} \pi$, then there is t_ρ such that $s_\rho \to t_\rho$ and $(\mathbb{FM}(\mathcal{TKS}^\rho), \to, t_\rho) \approx \mathsf{TM}_\pi$.*

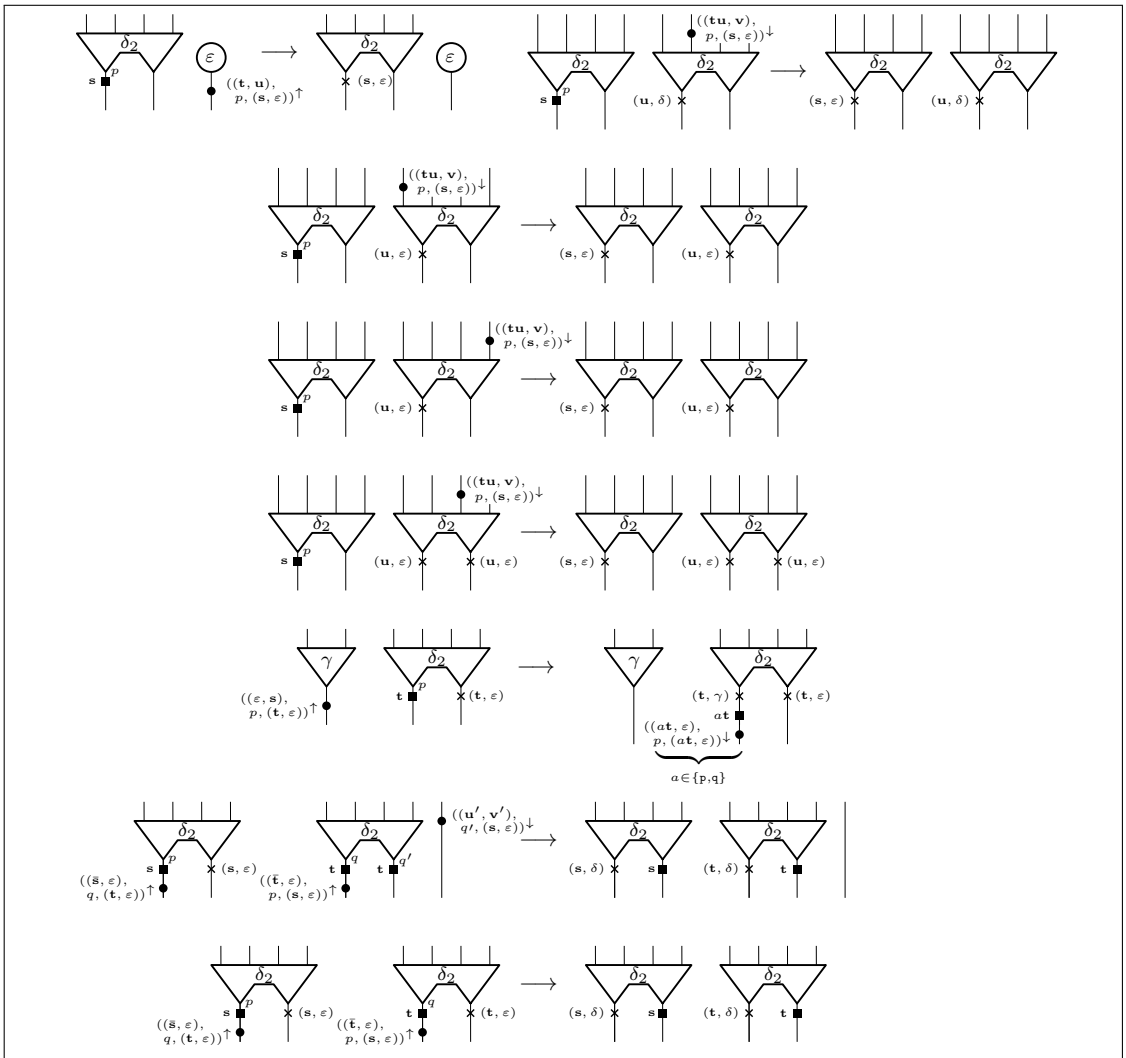


Figure 16: Internal Marriage Rules with ε

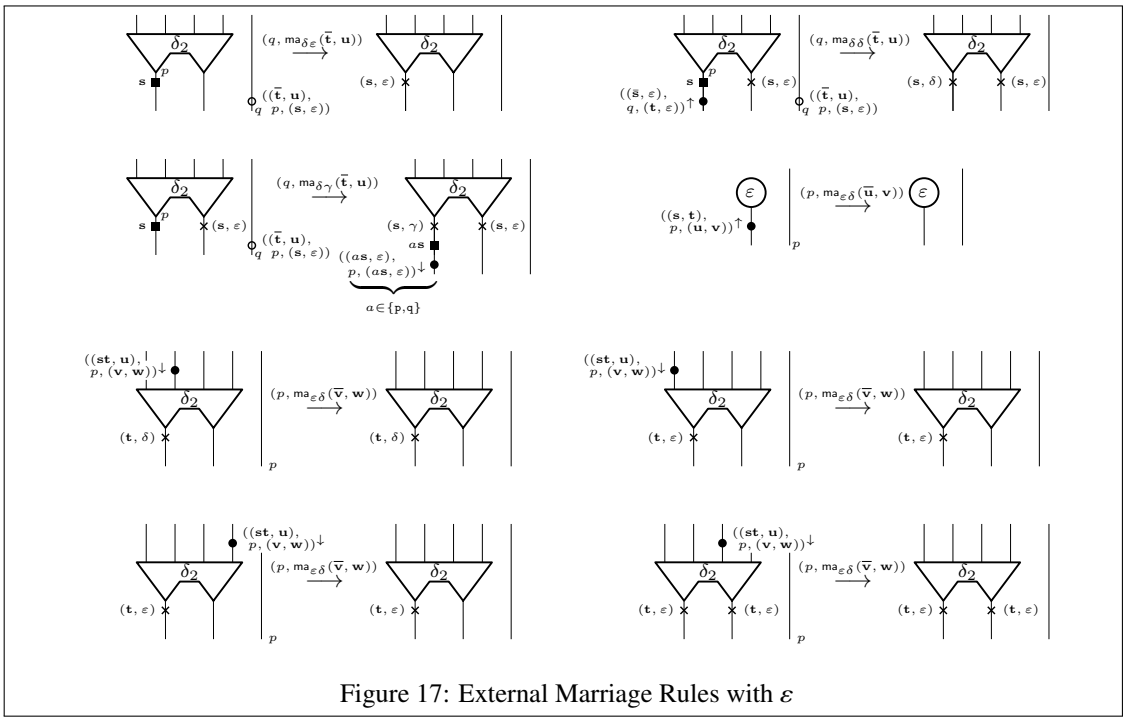
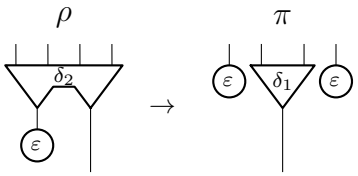


Figure 17: External Marriage Rules with ε

Proof. We are in the following situation:



Let t_ρ be a state which is obtained by performing a $\delta\varepsilon$ marriage in s_ρ. Then we define a bisimulation relation in the same way as Lemma 8. □

Adequacy. Finally, also the adequacy proof needs to be adapted, but only as for some of the technical lemmas, the overall structure of the proof remaining essentially unaltered. In particular, whenever we repeatedly apply $\gamma\gamma$ in the proofs, we instead apply the rules $\{\gamma\gamma, \gamma\varepsilon, \varepsilon\varepsilon\}$. This set of rules is also confluent and terminating, because the number of cells (in particular γ cells) is decreasing.