A Petri Net Semantics for π-Calculus *

Nadia Busi[1] and Roberto Gorrieri[2]

[1] Dipartimento di Matematica, Università di Siena,
Via del Capitano 15, I-53100 Siena, Italy
e-mail: busi@cs.unibo.it

[2] Dipartimento di Scienze dell'Informazione, Università di Bologna,
Piazza di Porta S. Donato 5, I-40127 Bologna, Italy
e-mail: gorrieri@cs.unibo.it

Abstract. A distributed semantics for π-calculus, based on P/T Petri nets with inhibitor arcs, is presented. This net semantics is sound w.r.t. the original early transition system semantics: the interleaving semantics of an agent p is retrievable from the net by considering the interleaving marking graph for the associated marking $dec(p)$. Furthermore, π-calculus is equipped with a multistep and a causal semantics. The latter is compared with recent proposals appeared in the literature.

1 Introduction

Distributed semantics for CCS and related languages based on (different classes of) Petri Nets have received a lot of attention in recent years (see, e.g., [5, 8, 10, 19], just to mention a few).

The Place/Transition net semantics proposed in the literature can be classified into two main groups. The first one, we call *location*-oriented, exploits the syntactical structure of the process terms (notably, the parallel operator) to define their associated sets of places [5, 19]. The second one, we call *label*-oriented, ignores the syntactical structure of the parallel operator and keeps only the information on the label of the transitions [9, 11]. While the former approach has the merit of being very general and successfully applied to several process algebras, the latter has the merit of producing small(er) net representations. Conversely, the former usually produces rather huge (1-safe P/T) nets, while the latter cannot be easily extended to cope with certain operators, e.g., restriction and alternative composition (see [12] for a discussion on this topic).

In a recent paper [3], we have shown how the label-oriented approach can be extended in a natural way to CCS by exploiting *inhibitor arcs*. This extension is conservative in the sense that, on the sublanguage without restriction and summation, inhibitor arcs are never used. The main features of our proposal were:

(i) For each action a, there is a place named (νa) which, when containing at least one token, inhibits the execution of all the transitions with label a or $\bar{a}$.

* Research partially supported by EC BRA n. 9102 COORDINATION.

(*ii*) Each agent of the form $p + q$ is first translated into the term $(\{k\}p|\{\bar{k}\}q)$, where k is called a *conflict* name and $\bar{k}$ is its contrasting conflict name. Hence, the two alternative agents are considered as parallel agents "decorated" by contrasting conflict names. The net semantics contains a place for any conflict name k, which, when holding at least one token, inhibits all the transitions starting from places/agents decorated by $\bar{k}$ (and symmetrically for $\bar{k}$, with the assumption that $\bar{\bar{k}} = k$).

(*iii*) Moreover, as we do not consider the syntactical structure of parallel composition, we always need fresh names for actions to rename the restriction binders.

Hence, this net semantics implements restriction and alternative composition by checking whether the transition is inhibited by any restrictions or conflicts it might be subject to.

In this paper we show that our approach can be extended to process algebras for mobile processes, by studying the prominent case of π-calculus [16]. The main new problems one has to face are the following:

(*i*) the explicit presence of values for messages (which are actually names), and the fact that

(*ii*) restriction is a *dynamic* operator which changes the scope of application as the computation proceeds.

Both aspects are easily accommodated in our approach, yielding the first – to the best of our knowledge – net-oriented semantics for full π-calculus. Note that the location oriented approach is not easily extensible to π-calculus because of extrusion: to perform a bound output of a channel y, all the sequential subprocesses inside the scope of (νy) need to be updated. This can be obtained by an explicit synchronisation of all these subprocesses, thus introducing possibly wrong causes.

We claim that our net semantics is simple because:

(*i*) Its definition is very concise if compared with the original one, based on transition systems. As a matter of fact, only five axiom schemata are needed to generate the whole set of net transitions. Moreover, they are very much in the style of the CHAM [1], where no induction on the syntactic structure needs to be performed.

(*ii*) The intuition behind the net semantics is immediate, once one has understood the role of inhibitors. Indeed, our semantics clearly reflects the "inhibiting role" played by restriction and the nature of distributed choice implemented through inhibiting conflict names.

(*iii*) It provides concise net descriptions of both distributed choice (*linear* in the size of the components, as discussed in [3]) and restriction (by not forcing to interpret parallel composition as *disjoint* union on places).

We prove a *retrievability* result: the interleaving marking graph associated to the net for π-calculus tightly corresponds to its *early* interleaving transition system.

As far as non-interleaving semantics are concerned, we observe that nets with inhibitor arcs have not been deeply studied: only some contrasting proposals are reported in the literature [14, 18]. We adapt one of these [18] to P/T nets which is suited for our aims, hence providing π-calculus with a multistep semantics (new – to the best of our knowledge). Finally, we present a causal semantics for P/T nets with inhibitor arcs; the induced causal semantics for π-calculus is compared with those recently appeared in the literature [2, 6].

1.1 Related Work

Net Semantics
The only paper we know dealing with a net semantics is [7]. There Engelfriet considers the small π-calculus (no alternative composition), for which he studies the reduction semantics (communications only). Because of these limited aims, the semantics is greatly simplified: the restriction operator can always be syntactically removed by simply renaming bound actions to fresh names. Hence, there is no need for inhibitors. It is easy to see that our net semantics conservatively extends his proposal.[3]

Causal Semantics
The first causal semantics was proposed in [2], where Boreale and Sangiorgi define a causal transition system in the style of [4]. They observe that there are two different sources of causality: the structural (or subject) dependencies due to the syntactical structure of terms (causes produced by the nesting of prefixing or inherited through communications) and the link (or object) dependencies due to extrusion, which enables transitions which were previously blocked by the presence of restriction. However, only the class of structural causes is explicitly described in the transition system. Our causal semantics, instead, joins together the causes originated by both classes of phenomena. Consider the two processes $\boldsymbol{p} = (\nu y)(\bar{x}y.\mathbf{0} \,|\, \bar{y}z.\mathbf{0})$ and $\boldsymbol{q} = (\nu y)(\bar{x}y.\bar{y}z.\mathbf{0})$. These processes are not causal bisimilar according to [2] (there is a link dependency in the former agent), whereas they are causal bisimilar for us.

Our approach to causality is also different from the one given in [6]. First of all, they follow the so-called *read-write* causality, according to which outputs can pass their causes to inputs but not viceversa (i.e., the cause-crossing in communication is not symmetric but directioned from outputs to inputs). The "symmetric" causes missing in communication are classified as structural priority. Similarly for link dependencies, they define object priority. We conjecture that the reflexive closure of our causal semantics coincides with the enabling relation of [6], which roughly corresponds to the transitive and reflexive closure of the union of the four relations above.

In [17] Montanari and Pistore propose a graph–rewriting system (based on the double pushout approach) as a semantic model for a subset of π-calculus,

[3] To be precise, this is true with the only exception of the bang operator, as we prefer to use guarded CCS-like recursion in order to avoid the problem of managing markings composed by infinite places and/or infinite tokens in a place (see the concluding section for more details).

where $+$ is omitted and the bang operator is applied only to input prefixing. Apart from the technical differences, the basic intuition about restriction is similar to ours, even if "complemented": a "token" in a "place" for x means that x is free; hence they test for presence of a token in the "place" for x. However, this causes the debatable phenomenon of parallel extrusion. Consider the process $p = (\nu y)(\bar{x}y.\mathbf{0} \mid y(w).\mathbf{0} \mid \bar{z}y.\mathbf{0})$: the two actions $\bar{x}(y)$ and $\bar{z}(y)$ are considered independent and executable in parallel in [17], while in interleaving semantics only one of the two outputs is actually a bound output.

The last paper is [15], where a data-flow semantics for π-calculus, based on dI-domains, is proposed. As the notion of causality they use is classical, we claim that it coincides with our one.

2 The π-Calculus

We recall the syntax and the interleaving semantics of (our variant of) the π-calculus [16].

Let $\mathcal{N}$ be a denumerable set of names, ranged over by $a, b, \ldots, x, y, \ldots$, such that $\tau \notin \mathcal{N}$. Let $\mathcal{X}$ be a denumerable set of process variables, disjoint from $\mathcal{N} \cup \{\tau\}$, ranged over by $X, Y, Z, \ldots$. The terms are generated from names by the following grammar:

$$p ::= \mathbf{0} \mid \mu.p \mid p + p \mid p \,|\, p \mid (\nu x)p \mid X \mid recX.p$$

where the prefix μ can be either an input $x(y)$, an output $\bar{x}y$ or a silent move τ.

The set $\mathcal{P}$ of *processes* contains those terms which are *closed* (w.r.t. process variables) and *guarded*. $\mathcal{P}$ will be ranged over by $p, q, \ldots$. A process of the form $\mu.p$ is called *sequential*. The sets of names, free names and bound names of a process p, called $n(p)$, $fn(p)$ and $bn(p)$ respectively, are defined as usual. The definitions of substitution and alpha conversion are standard, with the use of renaming to avoid name clashes.

The (early) operational semantics for the π-calculus is the labelled transition system generated by the rules listed in Table 1. The labels of the transitions are of four different kinds: the silent action τ, the input xy, the free and bound outputs $\bar{x}y$ and $\bar{x}(y)$. The set of labels is called $\mathcal{A}$ and is ranged over by α. The free and bound names of an action α, $fn(\alpha)$ and $bn(\alpha)$, are defined as follows: $fn(\tau) = bn(\tau) = \emptyset$, $fn(\bar{x}y) = fn(xy) = \{x, y\}$, $bn(\bar{x}y) = bn(xy) = \emptyset$, $fn(\bar{x}(y)) = \{x\}$, $bn(\bar{x}(y)) = \{y\}$.

The names of α are $n(\alpha) = fn(\alpha) \cup bn(\alpha)$.

Definition 1. A binary relation R over the set of terms is an *early bisimulation* if $(p, q) \in R$ implies:

- if $p \xrightarrow{\alpha} p'$ and $bn(\alpha) \cap fn(p, q) = \emptyset$ there exists then q' such that $q \xrightarrow{\alpha} q'$ and $(p', q') \in R$;
- symmetrically for q derivations.

Two terms p and q are *bisimilar*, written $p \sim q$, if there exists a bisimulation R such that $(p, q) \in R$. □

(Tau)	$\tau.p \xrightarrow{\tau} p$	
(In)	$x(z).p \xrightarrow{xy} p\{y/z\}$	
(Out)	$\bar{x}y.p \xrightarrow{\bar{x}y} p$	
(Sum)	$\dfrac{p \xrightarrow{\alpha} p'}{p+q \xrightarrow{\alpha} p'}$	
(Par)	$\dfrac{p \xrightarrow{\alpha} p'}{p \mid q \xrightarrow{\alpha} p' \mid q}$	$bn(\alpha) \cap fn(q) = \emptyset$
(Com)	$\dfrac{p \xrightarrow{\bar{x}y} p' \qquad q \xrightarrow{xy} q'}{p \mid q \xrightarrow{\tau} p' \mid q'}$	
(Close)	$\dfrac{p \xrightarrow{\bar{x}(w)} p' \qquad q \xrightarrow{xw} q'}{p \mid q \xrightarrow{\tau} (\nu w)(p' \mid q')}$	$w \notin fn(q)$
(Res)	$\dfrac{p \xrightarrow{\alpha} p'}{(\nu y)p \xrightarrow{\alpha} (\nu y)p'}$	$y \notin n(\alpha)$
(Open)	$\dfrac{p \xrightarrow{\bar{x}y} p'}{(\nu y)p \xrightarrow{\bar{x}(w)} p'\{w/y\}}$	$y \neq x \wedge w \notin fn((\nu y)p')$
(Rec)	$\dfrac{p\{\mathrm{rec}\, X.p/X\} \xrightarrow{\alpha} p'}{\mathrm{rec}\, X.p \xrightarrow{\alpha} p'}$	

Table 1. early semantics (symmetric rules omitted)

3 P/T Nets with Inhibitor Arcs

We recall simple Place/Transition nets without capacity constraints on places (see, e.g., [10]). Then, we extend them with the so-called inhibitor arcs (see, e.g., [13, 14, 18]). Here we provide a characterization of this model which is convenient for our aims.

Definition 2. Given a set S, a *finite multiset* over S is a function $m : S \to \omega$ such that the set $dom(m) = \{s \in S \mid m(s) \neq 0\}$ is finite. The *multiplicity* of an element s in m is given by the natural number $m(s)$. The set of all finite multisets over S, denoted by $\mathcal{M}_{fin}(S)$, is ranged over by m. A multiset m such that $dom(m) = \emptyset$ is called *empty* . The set of all finite sets over S is denoted by

$\wp_{fin}(S)$. If A is a finite subset of S, sometimes with abuse of notation we use A to denote the multiset m_A defined as follows: $m_A(s) =$ if $s \in A$ then 1 else 0.

We write $m \subseteq m'$ if $m(s) \leq m'(s)$ for all $s \in S$. The operator $\oplus$ denotes *multiset union*: $m \oplus m'(s) = m(s) + m'(s)$. The operator $\setminus$ denotes *multiset difference*: $m \setminus m'(s) = m(s) \dot{-} m'(s)$ where $i \dot{-} j = i - j$ if $i > j$, $i \dot{-} j = 0$ otherwise. Finally, the *scalar product*, $j \cdot m$, of a number j with a multiset m is $(j \cdot m)(s) = j \cdot (m(s))$. $\square$

Let f be a function on S. The function is extended to multiset over S in the following way: if $m \in \mathcal{M}_{fin}(S)$ then $(f(m))(s) = \sum_{f(\bar{s})=s} m(\bar{s})$.

Definition 3. A P/T net is a tuple $N = (S, Act, T)$ where S is the set of *places*, Act is the set of *labels* and $T \subseteq \mathcal{M}_{fin}(S) \times Act \times \mathcal{M}_{fin}(S)$ is the set of *transitions*. A finite multiset over the set S of places is called a *marking*. Given a marking m and a place s, we say that the place s contains $m(s)$ *tokens*.

A transition $t = (c, a, p)$ is usually written in the form $c \xrightarrow{a} p$. The marking c, usually denoted with ${}^\bullet t$, is called the *preset* of t and represents the tokens to be "consumed"; the marking p, usually denoted with $t^\bullet$, is called the *postset* of t and represents the tokens to be "produced"; a is called the *label* of t, sometimes denoted by $l(t)$. A transition t is *enabled* at m if ${}^\bullet t \subseteq m$. The execution of a transition t enabled at m produces the marking $m' = (m \setminus {}^\bullet t) \oplus t^\bullet$. This is written as $m[t\rangle m'$.

A *marked* P/T net is a tuple $N(m_0) = (S, Act, T, m_0)$, where (S, Act, T) is a P/T net and m_0 is a nonempty finite multiset over S, called the *initial marking*. $\square$

Definition 4. A P/T net *with inhibitor arcs* (NI net for short) is a tuple $N = (S, Act, T)$ where S is the set of places, Act the set of labels and $T \subseteq \mathcal{M}_{fin}(S) \times \wp_{fin}(S) \times Act \times \mathcal{M}_{fin}(S)$ is the set of transitions.

A transition $t = (c, i, a, p)$ is usually written in the form $(c, i) \xrightarrow{a} p$. The set i, denoted with ${}^\circ t$, is called the *inhibitor-set* of t and represents the tokens to be "tested for absence"; markings c and p are as above. This changes the definition of enabling: a transition t is enabled at m if ${}^\bullet t \subseteq m$ and $dom(m) \cap {}^\circ t = \emptyset$. Thus, any transition t for which ${}^\circ t \cap dom({}^\bullet t) \neq \emptyset$ is called *blocked*, *unblocked* otherwise.

The execution of a transition t enabled at m producing the marking m', written $m[t\rangle m'$, is defined as above. $\square$

Definition 5. Let $N = (S, Act, T)$ be a net. The *interleaving marking graph* of N is $IMG_N = (\mathcal{M}_{fin}(S), \rightarrow, Act)$, where $\rightarrow \subseteq \mathcal{M}_{fin}(S) \times Act \times \mathcal{M}_{fin}(S)$ is defined by $m \xrightarrow{l(t)} m'$ iff there exists a transition $t \in T$ such that $m[t\rangle m'$. With $IMG_N(m)$ we denote the interleaving marking graph reachable from the initial marking m. $\square$

4 Net Semantics

Let $\mathcal{C}$ be a denumerable set of symbols disjoint from $\mathcal{N} \cup \{\tau\}$ and from $\mathcal{X}$. Let $\bar{\mathcal{C}} = \{\bar{k} \mid k \in \mathcal{C}\}$ and $Con = \mathcal{C} \cup \bar{\mathcal{C}}$. Con is ranged over by $k, h, \ldots$, with the assumption that $\bar{\bar{k}} = k$. A *conflict set* is a finite subset of $Con = \mathcal{C} \cup \bar{\mathcal{C}}$, ranged over by $I, J, \ldots$; $\bar{I} = \{\bar{k} \mid k \in I\}$. When I is a singleton, we drop the set brackets for notational convenience. The set S_π of places is defined as follows:

$$S_\pi \;=\; \{I[\mu.p]_{\equiv_\alpha} \mid I \subseteq Con \wedge \mu.p \in \mathcal{P}\} \;\cup\; Con \;\cup\; \{(\nu x) \mid x \in \mathcal{N}\}$$

$I\mu.p$ represents a sequential process with conflict set I. This set, sometimes omitted when empty, is essential for the implementation of the distributed choice mechanism, as will be made clear in the following. Sequential processes which are alpha convertible are identified. This abstraction step is not necessary for the technical development, but it makes net description more concise. For instance, processes $x(y).(\nu z)\bar{y}z$ and $x(v).(\nu w)\bar{v}w$ are mapped to the same place $\emptyset[x(y).(\nu z)\bar{y}z]_{\equiv_\alpha}$. For notational convenience, we usually write $I\mu.p$ for $I[\mu.p]_{\equiv_\alpha}$.

Every conflict name k is a place (with the same name). k is used to prevent the execution of any transition from any sequential process $I\mu.p$ such that $\bar{k} \in I$. To better describe the implementation of the distributed choice, assume that p and q are sequential processes; $p+q$ is interpreted as the parallel composition of $\{k\}p$ and $\{\bar{k}\}q$; with the execution of an action from $\{k\}p$ we produce a token in place k, hence preventing a later execution of an action from $\{\bar{k}\}q$.

Finally, observe that for any name x there is a place (νx). A token in such a place prevents the execution of input, output and bound output actions along channel x, as well as to send x as a free value or to receive x; however, it has no influence on synchronizations.

In order to define the decomposition function dec which associates a finite multiset on S_π to each process, we need some auxiliary definitions.

The operator $I(\_)$ is defined on places as follows: $I(J\mu.p) = (I \cup J)\mu.p$, $I(k) = k$ and $I((\nu x)) = (\nu x)$.

The renaming is defined on places as follows: $(I\mu.p)\{y/x\} = I(\mu.p)\{y/x\}$, $k\{y/x\}$
$= k$, $(\nu x)\{y/x\} = (\nu y)$ and $(\nu z)\{y/x\} = (\nu z)$ if $z \neq x$.

The renaming of conflicts is defined on places as follows: $h\{k/h\} = k$, $\bar{h}\{k/h\} = \bar{k}$, $h'\{k/h\} = h'$ if $h' \neq h, \bar{h}$, $I\{k/h\} = \{h'\{k/h\} \mid h' \in I\}$, $(I\mu.p)\{k/h\} = I\{k/h\}\mu.p$ and $(\nu x)\{k/h\} = (\nu x)$.

The set $c(s)$ of *conflicts of place* s is defined as: $c(I\mu.p) = \{k \in \mathcal{C} \mid k \in I \cup \bar{I}\}$, $c(k) =$ if $k \in \mathcal{C}$ then $\{k\}$ else $\{\bar{k}\}$, $c((\nu x)) = \emptyset$ and extended to markings as: $c(m) = \bigcup_{s \in dom(m)} c(s)$.

The set $n(s)$ of *names of place* s is defined as: $n(I\mu.p) = n(\mu.p)$, $n(k) = \emptyset$ and $n((\nu x)) = \{x\}$ and extended to markings as: $n(m) = \bigcup_{s \in dom(m)} n(s)$.

The set $r(m)$ of *restricted* names in marking m is defined as follows: $r(m) = \{x \mid (\nu x) \in dom(m)\}$.

For the sake of simplicity, we often write the functions above with multiple arguments, meaning the union of the applications to each argument. E.g., $r(m, m')$ means $r(m) \cup r(m')$.

We define a notion of alpha conversion on markings, which states that the conflicts and restricted names actually used are inessential.

Definition 6. The binary relation $\equiv_\alpha^1$ on markings is defined as the least relation generated by the two rules below:

(i) If $k, \bar{k} \notin c(m)$, then $m\{k/h\} \equiv_\alpha^1 m$.
(ii) Let $x \in r(m)$. If $w \notin n(m)$, then $m\{w/x\} \equiv_\alpha^1 m$.

Alpha conversion on markings, denoted $\equiv_\alpha$ with abuse of notation, is the reflexive and transitive closure of $\equiv_\alpha^1$. I.e., $m \equiv_\alpha m'$ if $m(\equiv_\alpha^1)^* m'$. □

$dec(\mathbf{0}) = \emptyset$	
$dec(\mu.p) = \{\emptyset\mu.p\}$	
$dec(p+q) = k dec(p) \oplus \bar{k} dec(q)$	k new
$dec(p \mid q) = dec(p) \oplus dec(q)$	
$dec((\nu x)p) = dec(p\{z/x\}) \oplus \{(\nu z)\}$	z new
$dec(\mathrm{rec}\, X.p) = dec(p\{\mathrm{rec}\, X.p/X\})$	

Table 2. Decomposition function

The function $\boldsymbol{dec} : \mathcal{P} \to \mathcal{M}_{fin}(S_\pi)$, which defines the decomposition of processes into markings, is reported in Table 2. Agent $\mathbf{0}$ generates no places. The decomposition of $\mu.p$ produces one place, where $\mu.p$ has an empty set of conflicts. The case of alternative composition is interesting, as it shows that $p+q$ is turned into the multiset union of the markings for p and q, where each place of $\boldsymbol{dec}(p)$ is decorated by the singleton $\{k\}$ and, symmetrically, each place in $\boldsymbol{dec}(q)$ is decorated by $\{\bar{k}\}$. Parallel composition is interpreted as multiset union. The decomposition of a restricted process $(\nu x)p$ generates a token in a restriction place (νz) (for a new name z) and the multiset obtained from the decomposition of p where the new name z is substituted for the bound name x. Finally, a recursive process is first unwounded once and then decomposed. Note that function $\boldsymbol{dec}$ is defined up to alpha conversion of markings, because the actual choice of the *new* conflict k and of the *new* name z is inessential.

The NI net for π-calculus is the triple $N_\pi = (S_\pi, \mathcal{A}, T_\pi)$, where the set T_π of net transitions is the least set generated by the axiom schemata reported in Table 3.

Axiom (tau) is trivial: if no token is present in the places of the contrasting conflict names, then the transition produces, besides the tokens of $\boldsymbol{dec}(p)$, also one token in each place of the conflicts in I. In this way, any other sequential process decorated with a conflict in $\bar{I}$ is blocked forever. Axiom (in) is also very easy: besides the contrasting conflict names, there is to check the absence of tokens on the restriction place for the channel name x and for the received value y. Axiom (out) is very similar to the above. Axiom (open) clearly reveals the essence of the bound output: it differs from (out) because the token in the restriction place (νy) is now to be consumed. Indeed, the effect of the extrusion of the name y is the removal of the restriction on y. Finally, axiom (sync) describes the interaction: if the two sequential processes are *compatible* (i.e., there are no contrasting conflicts, as required by the side-condition $I \cap \bar{J} = \emptyset$), then the communication can take place. Note that axioms (tau) and (sync) are the only two which do not make any use of the restriction places. Note also that a *reduction* semantics for full π-calculus can be described via axiom (sync) only. Finally, observe that conflicts are never consumed by transitions.

(tau)	$(\{I\tau.p\}, \bar{I}) \xrightarrow{\tau} dec(p) \oplus I$
(in)	$(\{Ix(z).p\}, \bar{I} \cup \{(\nu x), (\nu y)\}) \xrightarrow{xy} dec(p\{y/z\}) \oplus I$
(out)	$(\{I\bar{x}y.p\}, \bar{I} \cup \{(\nu x), (\nu y)\}) \xrightarrow{\bar{x}y} dec(p) \oplus I$
(open)	$(\{I\bar{x}y.p, (\nu y)\}, \bar{I} \cup \{(\nu x)\}) \xrightarrow{\bar{x}(y)} dec(p) \oplus I$
(sync)	$(\{Ix(y).p, J\bar{x}z.q\}, \bar{I} \cup \bar{J}) \xrightarrow{\tau} dec(p\{z/y\}) \oplus dec(q) \oplus I \oplus J$ if $I \cap \bar{J} = \emptyset$

Table 3. Axiom schemata for (early) transitions

Example 1. Here we provide a few examples which may help in better understanding the net semantics.

- Transition $(\{\{k\}x(y).\mathbf{0}, \{\bar{k}, (\nu x), (\nu z)\}) \xrightarrow{xz} \{k\}$ is one of the two transitions firable from the marking $dec(x(y).\mathbf{0} + v(w).\mathbf{0})$. If it is fired, there is no way to fire the other transition, because of the generated token in the place k.
- Consider process $(\nu y)(\bar{x}y.p|x(z).\mathbf{0})$, which is mapped to the marking $\{(\nu w), \bar{x}w.p, x(z).\mathbf{0}\}$. There are three possible enabled transitions from this marking, one for the asynchronous execution of the bound output of the left subprocesses, one for the asynchronous execution of an input from the right subprocess, and one for the synchronisation. The bound output transition is $(\{\bar{x}w.p, (\nu w)\}, \{(\nu x)\}) \xrightarrow{\bar{x}(w)} dec(p)$. Instead the synchronisation is $(\{\bar{x}w.p,$

$x(z).\mathbf{0}\}, \emptyset) \xrightarrow{\tau} dec(p)$. It is interesting to notice that for synchronisation the restriction places play no role.

– Transition $(\{\bar{x}x.\mathbf{0}, (\nu x)\}, \{(\nu x)\}) \xrightarrow{\bar{x}(x)} \emptyset$ is derivable by axiom (open), but it is blocked, because it needs to consume a token from a place where the presence of a token inhibits the transition itself. In the interleaving semantics, instead, a transition from $(\nu z)\bar{z}z.\mathbf{0}$ is not derivable, because of the side condition $x \neq y$ in rule (Open). $\square$

Given a process p, the NI net associated to p is the subnet of N_π reachable from the initial marking $dec(p)$. We can extend the definition of bisimulation on markings, provided that we devote some care to the treatment of bound names in our net semantics.

Definition 7. A binary relation R over the set of markings is an *early bisimulation* if $(m_1, m_2) \in R$ implies:

– if $m_1 \xrightarrow{\bar{x}y} m_1'$, there exists then m_2' such that $m_2 \xrightarrow{\bar{x}y} m_2'$ and $(m_1', m_2') \in R$;
– if $m_1 \xrightarrow{xy} m_1'$ and $y \notin r(m_1, m_2)$, there exists then m_2' such that $m_2 \xrightarrow{xy} m_2'$ and $(m_1', m_2') \in R$;
– if $m_1 \xrightarrow{\bar{x}(y)} m_1'$, there exist then m_2' and z such that $m_2 \xrightarrow{\bar{x}(z)} m_2'$ and – for $w \notin n(m_1, m_2)$ – $(m_1'\{w/y\}, m_2'\{w/z\}) \in R$;
– if $m_1 \xrightarrow{\tau} m_1'$, there exists then m_2' such that $m_2 \xrightarrow{\tau} m_2'$ and $(m_1', m_2') \in R$;
– symmetrically for m_2 derivations.

Two markings m_1 and m_2 are *bisimilar*, written $m_1 \sim m_2$, if there exists a bisimulation R such that $(m_1, m_2) \in R$. $\square$

Proposition 8. *Let m, m' be markings. If $m \equiv_\alpha m'$ then $m \sim m'$.* $\square$

4.1 Retrievability

In this section, we briefly sketch the proof of retrievability of the early interleaving semantics for process p from the interleaving marking graph of the net N_π marked with the initial marking $dec(p)$. The explicit treatment of bound names makes the correspondence a bit complex. The next two theorems compare the transition system with the interleaving marking graph. Then, the final corollary states that our net translation is fully abstract w.r.t. early interleaving bisimulation equivalence.

Theorem 9. *Let p be a process.*

– *If $p \xrightarrow{\bar{x}y} p'$, then there exists m such that $dec(p) \xrightarrow{\bar{x}y} m$ and $m \sim dec(p')$.*
– *If $p \xrightarrow{xy} p'$ and $y \notin r(dec(p))$, then there exists m such that $dec(p) \xrightarrow{xy} m$ and $m \sim dec(p')$.*
– *If $p \xrightarrow{\bar{x}(y)} p'$, then there exists m and w such that $dec(p) \xrightarrow{\bar{x}(w)} m$ and $m \sim dec(p'\{w/y\})$.*

– *If* $p \xrightarrow{\tau} p'$*, then there exists* m *such that* $dec(p) \xrightarrow{\tau} m$ *and* $m \sim dec(p')$. □

Theorem 10. *Let* p *be a process.*

– *If* $dec(p) \xrightarrow{\alpha} m$ *and* α *is not a bound output, then there exists* p' *such that* $p \xrightarrow{\alpha} p'$ *and* $m \sim dec(p')$.
– *If* $dec(p) \xrightarrow{\bar{x}(w)} m$*, then for every* $y \notin fn(p)$ *there exists* p' *such that* $p \xrightarrow{\bar{x}(y)} p'$ *and* $m \sim dec(p'\{w/y\})$. □

Corollary 11. *Let* p *and* q *be processes. Then we have that* $p \sim q$ *if and only if* $dec(p) \sim dec(q)$. □

5 Non-interleaving Semantics

We study the multistep semantics and the causal semantics for P/T nets with inhibitor arcs, and the induced non-interleaving semantics on π-calculus, through some examples.

5.1 Multistep Semantics

For nets with inhibitor arcs, a transition may be considered to be *concurrent* to another only if the former neither produces nor consumes tokens inhibiting the latter and vice versa. In other words, if a marking enables both, they can be executed in either order. Formally, a subset T' of unblocked transitions is a *set of possibly concurrent transitions* (SPCT for short) if $\forall t_1, t_2 \in T'. t_1 \neq t_2$ we have that $dom(t_1{}^\bullet) \cap {}^\circ t_2 = \emptyset = dom({}^\bullet t_1) \cap {}^\circ t_2$. A transition t which produces some token in its inhibitor-set (i.e., $dom(t^\bullet) \cap {}^\circ t \neq \emptyset$) is called *not self-concurrent.*

It is easy to observe that the net N_π contains self-concurrent transitions only.

Definition 12. Given a NI $N = (S, Act, T)$, let $G \in \mathcal{M}_{fin}(T)$ be a finite multiset of transitions. G is *enabled* at *marking* m iff

(i) $dom(G)$ is a SPCT,
(ii) if $t \in dom(G)$ is not self-concurrent, then $G(t) = 1$,
(iii) $m_1 \subseteq m$, where $m_1 = \bigoplus_{t \in dom(G)} G(t) \cdot {}^\bullet t$, and
(iv) $m(s) = 0$ for all $s \in \bigcup_{t \in dom(G)} {}^\circ t$.

If G is enabled at m, G is called a *multistep* from m to m', where $m' = (m \setminus m_1) \oplus m_2$ with $m_2 = \bigoplus_{t \in dom(G)} G(t) \cdot t^\bullet$. This will be denoted by $m[G\rangle m'$. □

Definition 13. Let $N = (S, Act, T)$ be a net. Its *multistep marking graph* is $MMG_N \stackrel{\text{def}}{=} (\mathcal{M}_{fin}(S), \rightarrow, Act)$ where $\rightarrow \subseteq \mathcal{M}_{fin}(S) \times \mathcal{M}_{fin}(Act) \times \mathcal{M}_{fin}(S)$ is defined by $m \xrightarrow{L} m'$ iff $\exists G \in \mathcal{M}_{fin}(T). m[G\rangle m'$ and $L \in \mathcal{M}_{fin}(Act)$ is such that $\forall a \in Act. L(a) = \sum_{l(t)=a} G(t)$. With $MMG_N(m)$ we denote the multistep marking graph reachable from the initial marking m. □

Example 2. We present two examples, in order to clarify what cannot be performed in parallel.

- The process $\bar{a}x.\mathbf{0} + \bar{b}x.0$ is mapped on the marking $m = \{k\bar{a}x.\mathbf{0}, \bar{k}\bar{b}x.\mathbf{0}\}$. In m two transitions are enabled: $t_1 = (\{k\bar{a}x.\mathbf{0}\}, \{\bar{k}, (\nu a), (\nu x)\}) \xrightarrow{\bar{a}x} \{k\}$ and $t_2 = (\{\bar{k}\bar{b}x.\mathbf{0}\}, \{k, (\nu b), (\nu x)\}) \xrightarrow{\bar{b}x} \{\bar{k}\}$. On the contrary, the step $\{t_1, t_2\}$ is not enabled at m, because it is not a SPTC: $dom(t_1^\bullet) \cap {}^\circ t_2 = \{k\} \neq \emptyset$.
- The process $(\nu y)(\bar{x}y.\mathbf{0} \,|\, y(v).nil \,|\, \bar{z}y.\mathbf{0})$ is mapped onto $m = \{(\nu w), \bar{x}w.\mathbf{0}, y(v).\mathbf{0}, \bar{z}w.\mathbf{0}\}$. Two transitions are enabled: $t_1 = (\{\bar{x}w.\mathbf{0}, (\nu w)\}, \{(\nu x)\}) \xrightarrow{\bar{x}(w)} \emptyset$ and $t_2 = (\{\bar{z}w.\mathbf{0}, (\nu w)\}, \{(\nu z)\}) \xrightarrow{\bar{z}(w)} \emptyset$. The step $\{t_1, t_2\}$ is not enabled because ${}^\bullet t_1 \oplus {}^\bullet t_2 \not\subseteq m$. In [17] the concurrent execution of t_1 and t_2 is allowed. □

The step bisimulation between markings can be defined similarly to interleaving bisimulation, and it is finer than this.

5.2 Causal Semantics

Given a marked NI net, we show how to generate a causal tree [4] in a rather simple way. This gives us a means for talking about (strong) causal bisimulation for π-calculus.

The construction proceeds by first defining types of tokens: they are essentially a decoration to a token that records some information about the way it has been generated, namely the occurrence of the transition which produced it. A place does not contain simply a set of tokens, rather a multiset of token types, because each individual token remembers its origin. Then, we introduce a notion of configuration of a marked net which essentially defines three pieces of information. The first is the present marking of the net, where the tokens are decorated by their type/history. The second is a function which associates to each place the set of transitions which have consumed tokens from that place; it will be useful because we consider that if a transition t consumes a token from the inhibitor set of a transition t', then t causes t'. The third piece of information is a counter of occurrences of transitions.

Definition 14. Let $N = (S, Act, T, m_0)$ be a marked P/T net with inhibitor arcs. The set of *token types* is $\Theta = (T \times \omega^+) \cup \{*\}$, ranged over by θ, where (t, i) is the type of tokens produced by the i-th occurrence of transition t and $*$ is the type of tokens in the initial marking.

A *configuration* γ of a net is a triple (p, e, o), where

- $p : S \rightarrow \Theta \rightarrow \omega$ describes for each place the number of tokens of each type it contains.
- $e : S \rightarrow \wp(T \times \omega^+)$ describes for each place the set of (occurrences of) transitions which have consumed tokens from that place.
- $o : T \rightarrow \omega$ defines the number of occurrences of each transition.

The *initial configuration* of the net is $\gamma_0 = (p_0, e_0, o_0)$, where

$$p_0(s)(\theta) = \begin{cases} m_0(s) & \text{if } \theta = * \\ 0 & \text{otherwise} \end{cases} \qquad e_0(s) = 0 \qquad o_0(t) = 0 \qquad \square$$

With $\gamma[t^i, C > \gamma'$ we denote the firing of transition t from configuration γ to configuration γ'. Actually, it is the i-th time that transition t is fired. Set C records the immediate causes for the firing of this occurrence; its elements are occurrences of transitions.

Definition 15. The rule for the i-causal firing rule is as follows: $(p, e, o)[t^i, C > (p', e', o')$ if and only if

- $\exists p_1 \subseteq p$ such that for all $s \in S \;\; {}^\bullet t(s) = \sum_\theta p_1(s)(\theta)$
- $\forall s \in {}^\circ t \;\; \sum_\theta p(s)(\theta) = 0$
- $o(t) = i - 1$
- $C = C_1 \cup C_2$, where $C_1 = \{(t, i) \mid \exists s : p_1(s)(t, i) > 0\}$ and $C_2 = \bigcup_{s \in {}^\circ t} e(s)$
- $p' = (p \setminus p_1) \oplus p_2$, where $p_2(s)(\theta) = \begin{cases} t^\bullet(s) & \text{if } \theta = (t, i) \\ 0 & \text{otherwise} \end{cases}$
- $e'(s) = \begin{cases} e(s) \cup \{(t, i)\} & \text{if } s \in {}^\bullet t \setminus t^\bullet \\ e(s) & \text{otherwise} \end{cases}$
- $o'(u) = \begin{cases} i & \text{if } u = t \\ o(u) & \text{otherwise} \end{cases}$ $\square$

As in C we have only the immediate causes, we want to define a causal firing sequence where the causal labelling contains all the causes via transitive closure of the immediate causes.

Definition 16. A causal firing sequence is defined inductively as follows:

- γ_0 is a causal firing sequence
- if $\gamma_0 \xrightarrow{t_1^{i_1}, D_1} \gamma_1 \cdots \xrightarrow{t_n, D_n} \gamma_n$ is a causal firing sequence and $\gamma_n[t_{n+1}^{i_{n+1}}, C > \gamma_{n+1}$ is a i-causal firing then $\gamma_0 \xrightarrow{t_1^{i_1}, D_1} \gamma_1 \cdots \xrightarrow{t_n, D_n} \gamma_n \xrightarrow{t_{n+1}^{i_{n+1}}, D_{n+1}} \gamma_{n+1}$, where $D_{n+1} = C \cup \bigcup_{t_j^{i_j} \in C} D_j$, is a causal firing sequence. $\square$

Finally, we build a causal tree from the set of causal firing sequences, by ordering them by prefix.

Definition 17. The causal tree of a net $N = (S, Act, T, m_0)$ is $CT(N) = (V, A, f)$, where

- V is the set of causal firing sequences of N;
- $A \subseteq V \times V$, $A = \{(\sigma, \sigma \xrightarrow{t^i, D} \gamma) \mid \sigma \wedge \sigma \xrightarrow{t^i, D} \gamma$ are causal firing sequences$\}$
- $f : A \to Act \times \wp(\omega^+)$, $f(\sigma, \sigma \xrightarrow{t_n^{i_n}, D_n} \gamma_n) = (l(t), \{n - k \mid t_k^{i_k} \in D_n\})$ for $\sigma = \gamma_0 \xrightarrow{t_1^{i_1}, D_1} \gamma_1 \cdots \xrightarrow{t_{n-1}^{i_{n-1}}, D_{n-1}} \gamma_{n-1}$ $\square$

Example 3. The net for the process $(\boldsymbol{\nu y})(\boldsymbol{\bar{x}y.0} \,|\, \boldsymbol{\bar{y}z.0})$ is drawn in Figure 1. This net can perform a causal transition labelled $\xrightarrow{\bar{x}(w),\emptyset}$, followed by the transition labelled $\xrightarrow{\bar{w}z,\{1\}}$. According to our definition above, the second transition is caused by the first one, because the first removes the inhibiting token. Instead, in [2], the second transition is not caused by the first one, because they do not consider explicitly link dependencies. □

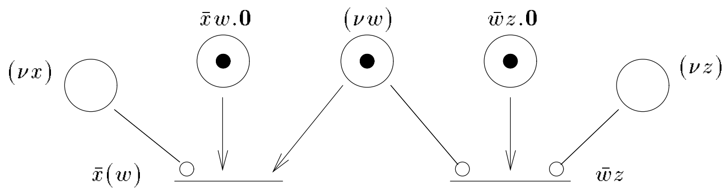

Fig. 1. The net for $(\nu y)(\bar{x}y.\mathbf{0} \,|\, \bar{y}z.\mathbf{0})$

The causal bisimulation between markings can be defined similarly to interleaving bisimulation, and it is finer than step bisimulation.

6 Conclusions

In [2, 6] it is considered important to distinguish the *structural* dependencies from the *link* dependencies. We want to stress that this kind of separation can be very naturally represented in our approach. Indeed, set C of immediate causes in Definition 15 is the union of two sets, C_1 and C_2. The first one considers all the causes due to the consumption of tokens produced by previously executed transitions. These are the causes due to flow arcs of the P/T net. Set C_2 contains, as causes of the present transition t, those transitions which have removed tokens from the inhibiting places of t. These are the causes due to the presence of inhibitor arcs. It is interesting to observe that the first set corresponds to the structural dependencies for π-calculus, while the second set expresses the link dependencies. So, a treatment of structural and link dependencies in our approach can be easily done by defining C as a pair (C_1, C_2).

References

1. G. Berry, G. Boudol, "The Chemical Abstract Machine", *Theoretical Computer Science* 96, 217-248, 1992.