ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **15**, Number 3, 2012, 171–214

Nominal Semantics of Mobility

Andrei ALEXANDRU, Gabriel CIOBANU

Romanian Academy, Institute of Computer Science, Iaşi
E-mail: aalexandru@iit.tuiasi.ro, gabriel@info.uaic.ro

Abstract. We present a nominal semantics of the π-calculus defined by a set of compact transition employing a nominal quantifier. We analyze the behavior of the π-calculus binding operators *new* and *input*, and several nominal properties of *new* and *input* are presented. A comparison of expressiveness between the nominal semantics of the π-calculus and other known semantics is made; the nominal semantics and the late semantics of the π-calculus have the same expressive power.

1. Introduction

The aim of this paper is to present a set of *compact* transition rules for the monadic version of the π-calculus. These transition rules are expressed using the quantifier $\forall$ and the nominal quantifier $\mathsf{И}$. The nominal quantifier helps us to "encode" the freshness conditions in the hypothesis of the transition rules, and to present a compact transition rule in the "weakest" form required to express mobility. Even if one transition rule in the new semantics of the π-calculus is assumed to be valid only when some freshness conditions are satisfied whilst its related rule in the late semantics of the π-calculus is assumed to be valid without requesting those freshness conditions to be satisfied, we are able to prove that the new semantics and the late semantics of the π-calculus are completely equivalent.

The notion of choosing a fresh name often arises when manipulating syntactic expressions; therefore it is necessary to indicate some constraints whenever describing such a syntactic manipulation. Often it is just said that a name is fresh without specifying any restrictions. In such a case, we mean that the fresh name must be different from any name occurring anywhere else in the expression or program. Some programming systems have mechanisms for renaming, for binding a name with a value and for managing sets of such bindings. Modern programming languages are designed to manage bindings and fresh names by using the notions of scope, workspace or

environment. Since renaming, binding and fresh names appear in several approaches, it became evident that they deserve to be studied in their own terms. The nominal logic and semantics was presented in [5, 6, 10]; it uses the Fraenkel-Mostowski(FM) model of set theory. The FM permutation model of set theory was devised in 1930s to prove the independence of the Axiom of Choice (AC) from the other axioms of Zermelo-Fraenkel (ZF) model of set theory. The axiom of choice says that given any collection of sets, each containing at least one object, it is possible to make a selection of exactly one object from each set, even though there are an infinite amount of sets and no "rule" of how we choose objects. As shown by Gödel and Cohen, the axiom of choice is logically independent of the other axioms of ZF model of set theory. The FM model is built using all the axioms of the Zermelo-Fraenkel with atoms (ZFA) model, except the axiom of choice. It has the special property of finite support which claims that for each element x in an arbitrary FM set we can find a finite set supporting x, as well as the property that the set A of atoms is infinite. In fact, the finite support property says that for each element x in an arbitrary FM set, we can always find a fresh element for x, i.e. an element which is not in the support of x. In λ-calculus, the α-equivalence classes of a λ-term x have the support of x represented exactly by the free variables with respect to x. This means that Fraenkel-Mostowski model of set theory could be a more suitable framework for computer science.

Some new formalisms have fresh operators (*e.g.*, operator *new* in process calculi), which allow the generation of a fresh name for a channel. Such a construction is guaranteed by the finite support property. Every freshness operator is also a binding operator, where the binding is an abstractive function in the Fraenkel-Mostowski model. The restriction $new\, xP$ in the π-calculus represents the local binding between the channel name x (which is assumed to be an *atom* in our construction) and the process P. The π-calculus was designed to be for the concurrent computation what is the λ-calculus for the sequential computation. Communication between processes in the π-calculus is realized by some communication channels, and parallel processes synchronize via message-passing handshakes on named channels. A benefit of the π-calculus is that the channels may be restricted (only certain processes may communicate on them). When a process sends a restricted name as a message to a process outside the scope of the restriction, the scope is said to extrude, that is, it enlarges to embrace the process receiving the channel. The communication possibilities of a process may change over time; a process may learn the names of new channels via scope extrusion. Thus, a channel is a transferable capability for communication.

The π-calculus embodies the view that interaction in distributed computation can be explained in terms of exchanges of names on named communication channels. The finite support property gives a mathematical reason for why such a renaming is always possible. Also, the Fraenkel-Mostowski set theory is the axiomatic support for the construction of nominal logic. Since we can use the same axiomatic model of set theory for describing both nominal logic and π-calculus, we apply nominal techniques in π-calculus.

In this paper we change the transition rules of the π-calculus by using α-abstraction defined by a freshness operator. For this we use a special nominal quantifier $И$ which provides the possibility of removing the free variables (which are represented by

the notion of finite support) from the scope of a rule. $И x$ means that x can be fresh for other parts of an expression; for example, $\forall x.И y.\forall z.expression$ is true iff $\forall x, y, z.(y\ is\, fresh\, for\, x) \Rightarrow (expression)$. Here we provide a consistent and new **nominal semantics** for the π-calculus. We prove several nominal properties of the binding operators of the π-calculus (namely *new* and *input*) which allow us to make a comparison between the nominal semantics of the π-calculus and the other known semantics defined before (*late* and *early* semantics). Using this new semantics, we study the mobility mechanism of the π-calculus given by links moving in a virtual space of linked processes. Even if the rules in the nominal semantics of the π-calculus look different than those in the late semantics of the π-calculus, the late semantics and the nominal semantics of the π-calculus have the same expressive power.

2. Fraenkel-Mostowski Set Theory and Nominal Logic

We consider the Fraenkel-Mostowski model with atoms by using the notions of transposition and permutation. Let A be an infinite set of atoms, characterized by the axiom "$y \in x \Rightarrow x \notin A$", which means that only non-atoms can have elements.

Definition 2.1.

i) A *transposition* is a function $(a\,b) : A \to A$ defined by
$$(a\,b)(a) = b,\ (a\,b)(b) = a \text{ and } (a\,b)(n) = n \text{ for } n \neq a, b.$$
ii) A *permutation* of A is generated by composing finitely many transpositions.

Let S_A be the set of all finitary permutations of A (*i.e.* the set of all permutations of A which leave unchanged all but finitely many atoms). S_A is a group under the usual composition of permutations. Actually S_A is a proper subgroup of the *symmetric* group on A (this is the reason why we chose to denote the set of finitary permutations on A by S_A). The composition of permutations is denoted by "$\circ$". It can be proved that S_A is the set of all bijections $\pi : A \to A$ generated by composing finitely many transpositions. Indeed, let $\sigma \in S_A$ be a function which permutes only a finite number of atoms $\{a_1, \ldots, a_n\}$ such that the atoms $A \setminus \{a_1, \ldots, a_n\}$ are left unchanged. Formally, we can say that σ is a permutation of the set $\{a_1, \ldots, a_n\}$, and so σ can be expressed as a product of at most $(n-1)$ transpositions [12].

Definition 2.2. Let X be a set defined by the axioms of ZF model. An S_A-*action* on X is a function $\cdot : S_A \times X \to X$ having the properties that $Id \cdot x = x$ and $\pi \cdot (\pi^{'} \cdot x) = (\pi \circ \pi^{'}) \cdot x$ for all $\pi, \pi' \in S_A$ and $x \in X$.

An S_A-*set* is a pair $(X, \cdot)$ where X is a set defined by ZF model, and $\cdot : S_A \times X \to X$ is an S_A-action on X. We simply use X whenever no confusion arises.

Definition 2.3. Let $(X, \cdot)$ be an S_A-set. We say that $S \subset A$ *supports* x whenever for each $\pi \in Fix(S)$ we have $\pi \cdot x = x$, where $Fix(S) = \{\pi \mid \pi(a) = a, \forall a \in S\}$.

When for an element x of an S_A-set we can find a finite set supporting it, we also say that "x has the finite support property" or "x is finitely supported".

Definition 2.4. Let $(X, \cdot)$ be an S_A-set. We say that X is a *nominal set* if for each $x \in X$ there exists a finite set $S_x \subset A$ which supports x.

Theorem 2.5. *Let X be an S_A-set, and for each $x \in X$ we define $\mathcal{F}_x = \{S \subset A \mid S \text{ finite, } S \text{ supports } x\}$. If $\mathcal{F}_x$ is nonempty then it has a least element which also supports x; this element is called* the support of x, *and it is denoted by $S(x)$ or $supp(x)$.*

Proof. We define $supp(x) = \cap\{S \subset A \mid S \text{ finite, } S \text{ supports } x\} = \underset{S \in \mathcal{F}_x}{\cap} S$. We prove that if S_1 and S_2 are both finite and supports x, then $S_1 \cap S_2$ supports x. Indeed, let π be a permutation from $Fix(S_1 \cap S_2)$. We prove that $\pi \cdot x = x$. Since any permutation π is generated by composing finitely many transpositions, it is enough to prove the finite support property of x only for transpositions. This means we must prove that for each $a, b \notin S_1 \cap S_2$ we have $(a\,b) \cdot x = x$. The cases $a, b \notin S_1$ and $a, b \notin S_2$ are obvious because S_1 and respectively S_2 supports x, and by Definition 2.3 we have $(a\,b) \cdot x = x$. Now let $a \notin S_1$ and $b \notin S_2$. Since $S_1 \cup S_2$ is finite and A is infinite, we can find $c \in A \setminus (S_1 \cup S_2)$ and $a \neq c \neq b$. Since $a, c \notin S_1$ and S_1 supports x, it follows that $(c\,a) \cdot x = x$. Because $c, b \notin S_2$ and S_2 supports x, we have $(c\,b) \cdot x = x$. It follows $(a\,b) \cdot x = (a\,b) \cdot (c\,b) \cdot x = ((a\,b) \circ (c\,b)) \cdot x = ((c\,b) \circ (a\,c)) \cdot x = (c\,b) \cdot (a\,c) \cdot x = x$. The case when $a \notin S_2$ and $b \notin S_1$ is similar. Let us suppose that $\mathcal{F}_x$ is nonempty. This means there is at least one finite set supporting x, and so the support $supp(x)$ is well defined. Moreover $supp(x)$ is minimal between the finite sets supporting x. □

Corollary 2.6. *Let X be a nominal set, and for each $x \in X$ let us define $\mathcal{F}_x = \{S \subset A \mid S \text{ finite, } S \text{ supports } x\}$. Then $\mathcal{F}_x$ has a least element which also supports x. We call this element* the support of x, *and we denote it by $S(x)$ or $supp(x)$.*

To avoid any possible confusion, we make the following remark: the symbols "S", "$supp(-)$" or "$S(-)$" are generally used for denoting sets supporting elements, while "S_A" denotes a subgroup of the symmetric group on A (S_A is the group of all finitary permutations of A).

Example 2.7.

1. *The set A of atoms is an S_A-set with the S_A-action $\cdot : S_A \times X \to X$ defined by $\pi \cdot a := \pi(a)$ for all $\pi \in S_A$ and $a \in A$. $(A, \cdot)$ is a nominal set because for each $a \in A$ we have that $\{a\}$ supports a. each $a \in A$.*

2. *The set A of atoms is an S_A-set with the S_A-action $\cdot : S_A \times X \to X$ defined by $\pi \cdot a := a$ for all $\pi \in S_A$ and $a \in A$. $(A, \cdot)$ is a nominal set because for each $a \in A$ we have that $\emptyset$ supports a. $supp(a) = \emptyset$ for each $a \in A$.*

3. *The set S_A is an S_A-set with the S_A-action $\cdot : S_A \times S_A \to S_A$ defined by $\pi \cdot \sigma := \pi \circ \sigma \circ \pi^{-1}$ for all $\pi, \sigma \in S_A$. $(S_A, \cdot)$ is a nominal set because for each $\sigma \in S_A$ we have that the finite set $\{a \in A \mid \sigma(a) \neq a\}$ supports σ.*

4. *Any ordinary ZF-set X (like $\mathbb{N}, \mathbb{Z}, \mathbb{Q}$ or $\mathbb{R}$ for example) is an S_A-set with the S_A-action $\cdot : S_A \times X \to X$ defined by $\pi \cdot x := x$ for all $\pi \in S_A$ and $x \in X$.*

Also X is a nominal set because for each $x \in X$ we have that $\emptyset$ supports x. $supp(x) = \emptyset$ for each $x \in X$.

5. *If $(X, \cdot)$ is an S_A-set then $\wp(X) = \{Y \mid Y \subseteq X\}$ is also an S_A-set with the S_A-action $\star : S_A \times \wp(X) \to \wp(X)$ defined by $\pi \star Y := \{\pi \cdot y \mid y \in Y\}$ for all permutations π of A, and all subsets Y of X. When no confusion arises we denote $\star$ also by $\cdot$. Note that $\wp(X)$ does not necessarily be a nominal set even if X is. For example A is a nominal set, but $\wp(A)$ is not a nominal set because the subsets of A which are in the same time infinite and coinfinite don't have the finite support property. For each nominal set $(X, \cdot)$ we denote by $\wp_{fs}(X)$ the set formed from those subsets of X which are finitely supported according to action $\star$. $(\wp_{fs}(X), \star|_{\wp_{fs}(X)})$ is a nominal set, where $\star|_{\wp_{fs}(X)} : S_A \times \wp_{fs}(X) \to \wp_{fs}(X)$ is defined by $\pi \star|_{\wp_{fs}(X)} Y := \pi \star Y$ for all $\pi \in S_A$ and $Y \in \wp_{fs}(X)$; the codomain of the action $\star|_{\wp_{fs}(X)}$ (which is the action $\star$ restricted to $\wp_{fs}(X)$) is indeed included in $\wp_{fs}(X)$ because of Proposition 2.8.*

6. *Let $(X, \cdot)$ and $(Y, \diamond)$ be S_A-sets. As in the classical ZF theory, we define the Cartesian product $X \times Y$ as the set of ordered pair $(x, y) = \{\{x\}, \{x, y\}\}$ for $x \in X$ and $y \in Y$. $X \times Y$ is also an S_A-set with the S_A-action $\star : S_A \times (X \times Y) \to (X \times Y)$ defined by $\pi \star (x, y) = (\pi \cdot x, \pi \diamond y)$ for all $\pi \in S_A$ and all $x \in X$, $y \in Y$. If $(X, \cdot)$ and $(Y, \diamond)$ are nominal sets, then $(X \times Y, \star)$ is also a nominal set.*

7. *Let $(X, \cdot)$ and $(Y, \diamond)$ be S_A-sets. We define the disjoint union of X and Y by $X + Y = \{(0, x) \mid x \in X\} \cup \{(1, y) \mid y \in Y\}$. $X + Y$ is an S_A-set with the S_A-action $\star : S_A \times (X + Y) \to (X + Y)$ defined by $\pi \star z = (0, \pi \cdot x)$ if $z = (0, x)$ and $\pi \star z = (1, \pi \diamond y)$ if $z = (1, y)$. If $(X, \cdot)$ and $(Y, \diamond)$ are nominal sets, then $(X + Y, \star)$ is also a nominal set: each $z \in X + Y$ is either of the form $(0, x)$ and supported by the finite set supporting x in X, or is of the form $(1, y)$ and supported by the finite set supporting y in Y.*

Proposition 2.8. *Let $(X, \cdot)$ be an S_A-set and let $\pi \in S_A$ be an arbitrary permutation. Then for each $x \in X$ which is finitely supported we have that $\pi \cdot x$ is finitely supported, and $supp(\pi \cdot x) = \pi(supp(x))$.*

Proof. Let $\pi \in S_A$ be an arbitrary permutation and $x \in X$ a finitely supported element. First we show that $\pi(supp(x))$ supports $\pi \cdot x$. Let $\sigma \in Fix(\pi(supp(x)))$, namely $\sigma(\pi(a)) = \pi(a)$ for all $a \in supp(x)$. Also $\pi^{-1}(\sigma(\pi(a))) = \pi^{-1}(\pi(a)) = a$, $\forall a \in supp(x)$. So we get $\pi^{-1} \circ \sigma \circ \pi \in Fix(supp(x))$. Now $supp(x)$ always supports x (by Corollary 2.6). According to Definition 2.3 we have $(\pi^{-1} \circ \sigma \circ \pi) \cdot x = x$. Since $\cdot$ is a group action, the last equality is equivalent with $\sigma \cdot (\pi \cdot x) = \pi \cdot x$. Hence whenever $x \in X$ is finitely supported, we have that $\pi \cdot x$ is finitely supported. Moreover $supp(\pi \cdot x) \subseteq \pi(supp(x))$ for each $x \in X$ which is finitely supported and each $\pi \in S_A$ (I).

We apply now (I) for elements $\pi^{-1} \in S_A$ and $\pi \cdot x \in X$ about which we already know that are finitely supported. We get $supp(\pi^{-1} \cdot \pi \cdot x) \subseteq \pi^{-1}(supp(\pi \cdot x))$. Composing with π in the last relation, we obtain $\pi(supp(x)) \subseteq supp(\pi \cdot x)$. □

As in [6], we can take a set-theoretic approach and construct a single 'large' S_A-set, *i.e.* an S_A-class (a class equipped with an S_A-action) denoted by $FM(A)$ whose all elements have the finite support property. One benefit is that if a particular construction can be expressed in this framework, then the action of permutations is inherited from the ambient universe $FM(A)$ without having to define it explicitly and without having to prove the associated finite support property. Recall now the usual von Neumann cumulative hierarchy of sets:

- $\nu_0 = \emptyset$
- $\nu_{\alpha+1} = \wp(\nu_\alpha)$
- $\nu_\lambda = \bigcup\limits_{\alpha \leq \lambda} \nu_\alpha$ (λ a limit ordinal).

More generally, given a set U, we can define analogue a cumulative hierarchy of sets involving atoms from U:

- $\nu_0(U) = \emptyset$
- $\nu_{\alpha+1}(U) = U + \wp(\nu_\alpha(U))$
- $\nu_\lambda(U) = \bigcup\limits_{\alpha \leq \lambda} \nu_\alpha(U)$ (λ a limit ordinal),

where $+$ denotes the disjoint union of sets defined in Example 2.7 (7). Let $\nu(U)$ be the union of all $\nu_\alpha(U)$. The class of sets built on atoms U is $\nu(U)$. We can build the notions of S_A-set and finite support property into such a hierarchy by tacking U to be the S_A-set A of atoms, and replacing $\wp(-)$ by $\wp_{fs}(-)$ (with the notations of Example 2.7):

- $FM_0(A) = \emptyset$
- $FM_{\alpha+1}(A) = A + \wp_{fs}(FM_\alpha(A))$
- $FM_\lambda(A) = \bigcup\limits_{\alpha \leq \lambda} FM_\alpha(A)$ (λ a limit ordinal).

According to Example 2.7, each $FM_\alpha(A)$ is a nominal set. When we consider the union of all $FM_\alpha(A)$, we get one 'large' S_A-set (*i.e.* an S_A-class) in which every element has finite support. The union of all $FM_\alpha(A)$ is called the *Cumulative Hierarchy Fraenkel-Mostowski (CHFM) universe*, and it is denoted by $FM(A)$. Using the names *atm* and *set* for the functions $x \mapsto (0, x)$ and $x \mapsto (1, x)$, respectively (the notations are preserved from Example 2.7), we have that every element x of $FM(A)$ is either of the form $atm(a)$ with $a \in A$, or of the form $set(X)$ where X is a finitely supported set formed at an earlier ordinal stage than x. We call *CHFM-sets* the elements of the form $set(X)$, and *atoms* the elements of the form $atm(a)$. S_A-action $\cdot$ on the CHFM universe $FM(A)$ can be defined recursively by

$$\pi \cdot atm(a) = atm(\pi(a)),\ \pi \cdot set(X) = set(\{\pi \cdot x \mid x \in X\}).$$

An element $x \in \nu(A)$ is an CHFM-set iff the following conditions are satisfied:

- y is a CHFM-set for all $y \in x$,
- x has finite support.

A CHFM-set x is not itself closed under the S_A-action on $FM(A)$ unless $supp(x) = \emptyset$. Hence a CHFM-set is not necessarily nominal set in the sense of Definition 2.4. We are especially interested on those CHFM-sets which are nominal sets. A nominal CHFM-set is a CHFM-set with empty support. Hence if X is a nominal CHFM-set, then the restriction of the S_A-action $\cdot$ on $FM(A)$ to $S_A \times X$ has the codomain equal with X because $\pi \cdot X = X$ for all $\pi \in S_A$. Nominal CHFM-sets are both S_A-sets and nominal sets.

The underlying logic of ZFA is the usual first-order logic with equality. Its signature contains just a binary predicate $\in$ denoting set membership, and a constant symbol A. We use the symbol "$\Rightarrow$" for "implies", and the symbol "$\Leftrightarrow$" for "if and only if".

Definition 2.9. The following axioms give a complete characterization of the Zermelo-Fraenkel model of set theory with atoms:

1. $\forall x.(\exists y.y \in x) \Rightarrow x \notin A$ (only non-atoms can have elements)
2. $\forall x, y.(x \notin A \, and \, y \notin A \, and \, \forall z.(z \in x \Leftrightarrow z \in y)) \Rightarrow x = y$ (axiom of extensionality)
3. $\forall x, y.\exists z.z = \{x, y\}$ (axiom of pairing)
4. $\forall x.\exists y.y = \{z \mid z \subset x\}$ (axiom of powerset)
5. $\forall x.\exists y.y \notin A \, and \, y = \{z \mid \exists w.(z \in w \, and \, w \in x)\}$ (axiom of union)
6. $\forall x.\exists y.(y \notin A \, and \, y = \{f(z) \mid z \in x\})$, *for each functional formula* $f(z)$ (axiom of replacement)
7. $\forall x.\exists y.(y \notin A \, and \, y = \{z \mid z \in x \, and \, p(z)\})$, *for each formula* $p(z)$ (axiom of separation)
8. $(\forall x.(\forall y \in x.p(y)) \Rightarrow p(x)) \Rightarrow \forall x.p(x)$ (induction principle)
9. $\exists x.(\emptyset \in x \, and \, (\forall y.y \in x \Rightarrow y \cup \{y\} \in x))$ (axiom of infinity)
10. *A is not finite.*

We present now the axioms of the Fraenkel-Mostowski model of set theory. This model is built on ZFA without choice axioms, having additionally a finite support property (axiom 11).

1. $\forall x.(\exists y.y \in x) \Rightarrow x \notin A$
2. $\forall x, y.(x \notin A \wedge y \notin A \wedge \forall z.(z \in x \Leftrightarrow z \in y)) \Rightarrow x = y$
3. $\forall x, y.\exists z.z = \{x, y\}$

4. $\forall x.\exists y.y = \{z \mid z \subset x\}$
5. $\forall x.\exists y.y \notin A \wedge y = \{z \mid \exists w.(z \in w \, and \, w \in x)\}$
6. $\forall x.\exists y.(y \notin A \wedge y = \{z \mid z \in x \, and \, p(z)\})$ for each formula $p(z)$
7. $\forall x.\exists y.(y \notin A \wedge y = \{f(z) \mid z \in x\})$ for each functional formula $f(z)$
8. $(\forall x.(\forall y \in x.p(y)) \Rightarrow p(x)) \Rightarrow \forall x.p(x)$
9. $\exists x.(\emptyset \in x \, and \, (\forall y.y \in x \Rightarrow y \cup \{y\} \in x))$
10. *A is not finite*,
11. $\forall x.\exists S \subset A.$ *S is finite and S supports* x. (finite support property)

It is easy to see that $\nu(A)$ is a model of ZFA set theory, and $FM(A)$ is a model of FM set theory. There exists some different approaches in the literature. Some authors define the Fraenkel-Mostowski sets (FM-sets) in the same way we define the nominal sets, whilst other authors define the Fraenkel-Mostowski sets as elements in the Fraenkel-Mostowski universe $FM(A)$ (built using the cumulative hierarchy presented before) which do not necessarily need to be nominal sets. In this paper we are interest especially on those elements in $FM(A)$ which are closed under the S_A-action on $FM(A)$.

We define an *FM-set* as an element in $FM(A)$ (*i.e.* as a CHFM-set). We define an *NFM-set* as a nominal set with a special S_A-action induced by the S_A-action on $FM(A)$. Precisely, an NFM-set is a set from ZFA which is closed under the S_A-action on $FM(A)$, and whose all elements are finitely supported. The S_A-action on an NFM-set are called *interchange function*.

Definition 2.10.

i) An element from the FM universe $FM(A)$ is called *Fraenkel-Mostowski set (FM-set)*.

ii) An *interchange function* on a set X defined by the axioms of ZFA model without choice is a function $\cdot : S_A \times X \to X$ defined inductively by $\pi \cdot a := \pi(a)$ for all atoms $a \in A$ and $\pi \cdot x := \{\pi \cdot y \mid y \in x\}$ which satisfies the axiom that for all $x \in X$ there is a finite subset $S \subset A$ such that $(a\,b) \cdot x = x$ for all $a, b \notin S$.

iii) An *NFM-set* is a pair $(X, \cdot)$, where X is a set defined by ZFA model without choice, and $\cdot : S_A \times X \to X$ is an interchange function on X.

Clearly, $FM(A)$ is an NFM-set.

Remark 2.11. *Since S_A is a group, the interchange function $\cdot : S_A \times X \to X$ is an action of group S_A on set X; we have $Id \cdot x = x$ and $\pi \cdot \pi^{'} \cdot x = (\pi \circ \pi^{'}) \cdot x$, $\forall \pi, \pi' \in S_A$. Thus we can express an NFM-set $(X, \cdot)$ like a set provided by an action $\cdot$ of S_A on X.*

Every NFM-set is also a nominal set. The converse is not valid.

Example 2.12. *The set of atoms A with the S_A-action defied as in Example 2.7 (1) is an NFM-set, whilst the set of atoms A with the S_A-action defied as in Example 2.7 (2) is a nominal set, but not an NFM-set.*

The property of the interchange function described in Definition 2.10 always allows one to find a finite set supporting x, for each element x in an arbitrary NFM-set. The following result follows from Corollary 2.6.

Theorem 2.13. *Let X be an NFM-set, and for each $x \in X$ we define $\mathcal{F}_x = \{S \subset A \mid S\ finite,\ S\ supports\ x\}$. Then $\mathcal{F}_x$ has a least element which also supports x. We call this element* the support of x, *and denote it by $S(x)$ or $supp(x)$.*

The following Example (considered also in [6]) shows us how we can express the usual λ-calculus in FM. Gabbay and Pitts have used this example to argue once more that we can use the FM approach instead of working in the classical ZF approach.

Example 2.14.

1. *If X' is the set of λ-terms, we define an action $\star$ of S_A on X' by:*
   - *variable: $\pi \star a = \pi(a)$ whenever a is a variable (corresponding to atoms) and π is a permutation of atoms;*
   - *application: $\pi \star (tt') = (\pi \star t)(\pi \star t')$ for all λ-terms t and t', and for all $\pi \in S_A$;*
   - *abstraction: $\pi \star (\lambda a.t) = \lambda(\pi(a)).(\pi \star t)$ for all variables a, all λ-terms t and for all $\pi \in S_A$.*

   It is easy to check that $(X', \star)$ is a nominal set (it is also an NFM-set by the definition of the S_A-action), and the support of a λ-term t is the finite set of atoms occurring in t, whether as free bound or binding occurrences.
2. *Let X be the set of the α-equivalences classes of the λ-calculus terms t. We can define an action $\cdot$ of S_A on X by: $\pi \cdot [t]_\alpha = [\pi \star t]_\alpha$ for all λ-terms t and all $\pi \in S_A$ (where $[t]_\alpha$represents the α-equivalence class of the λ-term t). If two λ-terms t and t' are α-equivalent, it is clear that $\pi \star t = \pi \star t'$, and so the action $\cdot$ is well defined. It is easy to check that $(X, \cdot)$ is a nominal set (it is also an NFM-set). Moreover X is a set which is in a bijection with an inductively defined FM-set (according to [6]). If t is chosen to be a representative of its α-equivalence class, then $supp(t)$ coincides with $fn(t)$, where $fn(t)$ is the set of free variables of t defined by λ-calculus rules [6]. An α-equivalence class of terms does not contain bound names (in the sense of the quotient of the equivalence class over them). We cannot define a function $bn : X \to \wp_{fin}(A)$ which would be able to extract exactly the bound names for each FM element t; α-equivalent terms are identified in the nominal logic since two α-equivalent terms have the same set of free variables.*

Definition 2.15. If $x \in A$ and $y \in Y$, where Y is a nominal set, we say that x *is fresh for* $supp(y)$ and denote this by $x\#y$ if $supp(x) \cap supp(y) = \emptyset$.

For an arbitrary name, we can always find a name outside its support, because for all y we know that $supp(y)$ is finite and, because A is infinite, we can find an atom x such that $x \notin supp(y)$. This means that $\forall x.\exists a \in A.a\#x$.

Remark 2.16. *If we suppose that $p((x_i)_i)$ is a formula in the logic of ZFA or FM, where the free variables of p are listed in the set $(x_i)_i$, and $(x_i)_i$ are arbitrary distinct*

variables, then by induction on the structure of p and by definition of the interchange function we obtain that $\forall a, b \in A.(p((x_i)_i) \Leftrightarrow p((a\,b)\cdot(x_i)_i))$*, where* $p((a\,b)\cdot(x_i)_i)$ *denotes the result of substituting* $(a\,b)\cdot x_j$ *for all free occurrences* x_j *from the set* $(x_i)_i$ *in p. This is an equivariance property. A complete proof of this property can be found in [6]. The idea of proving this is to proceed by induction on the structure of the FM formula p, using the properties:* $x = y \Rightarrow (a\,b)\cdot x = (a\,b)\cdot y$, $x \in y \Rightarrow (a\,b)\cdot x \in (a\,b)\cdot y$ *and* $(a\,b)\cdot A = A$*, where the last two properties follow from the definition of the interchange function. More information on equivariance in the FM and ZFA models can be found in [6].*

Definition 2.17. Let P be a predicate on A. We say that $\text{И}a.P(a)$ if $P(a)$ is true for all but finitely many elements of A. И is called the **nominal quantifier**.

This quantifier was first introduced in [6]. In that paper it is called "the new quantifier", and it is denoted by a reflected sans-serif roman letter N, by analogy with $\forall$ which is the reflected A of "for all" and $\exists$ the reflected E of "exists".

Remark 2.18. *Using its definition, it is easy to prove that the support of x can be expressed in the form:*

$$supp(x) = \{a \in A \mid \{b \in A \mid (a\,b)\cdot x \neq x\} \text{ is not finite}\},$$

and so $a \notin supp(x)$ *holds if and only if* $(a\,b)\cdot x = x$ *for all but finitely many* $b \in A$*. We write* $a\#x \Longleftrightarrow \text{И}b.(a\,b)\cdot x = x$*.*

Example 2.19. *Here we present some examples of how we can determine the support for various subsets of A:*

1. *If* $B \subset A$ *and B is finite, then* $supp(B) = B$*.*
2. *If* $C \subset A$ *and C is cofinite, then* $supp(C) = A \setminus C$*.*
3. *If* $D \subset A$ *is neither finite nor cofinite, then we cannot find a finite set supporting A. Indeed, let us assume that S supports D (this means that we have* $\pi(D) = D$ *for each* $\pi \in Fix(S)$*). If D is of form* $\{a, c, e, ...\}$ *(see Remark* **??***), then at least* $\{a, c, e, ...\}$ *or* $\{b, d, f, ...\}$ *(where* $\{b, d, f, ...\}$ *is* $C_D = A \setminus D$*) must be fixed by each* π*. This means that S cannot be finite. So we obtain* $\wp(A) = \wp_{fin}(A) \cup \wp_{cofin}(A)$ *in the FM set theory (where* $\wp_{fin}(A) = \{X | X \subset A,\ X \text{ finite}\}$*,* $\wp_{cofin}(A) = \{X | X \subset A,\ A \setminus X \text{ finite}\}$*). This means that we do not accept in the FM set theory the subsets of A which are at the same time infinite and coinfinite, because of the finite support property.*

Remark 2.20. *In Example 2.19,* $\{a, c, e, ...\}$ *is only a convention of representing D. We do not make a choice of atoms in the construction of D like: a is the first atom from A, c is the third atom from A, etc. We use the form* $\{a, c, e, ...\}$ *for D because, by intuition, it is easy to see what* C_D *is in this case. However the atoms composing D are arbitrarily presented in the structure of D, with no preliminary choice; the only condition is that both D and* C_D *should be infinite. For example, D can also be of form* $\{a, c, u, e, g, i, gh, ar_1, ...\}$*, where each of* $a, c, u, e, g, i, gh, ar_1, ...$ *are arbitrary atoms and both* $\{a, c, u, e, g, i, gh, ar_1, ...\}$ *and its complementary are infinite.*

The property of A which claims that $\wp(A) = \wp_{fin}(A) \cup \wp_{cofin}(A)$ is characteristic for the FM set theory. This structure of the set A of atoms allows us to say that *the axiom of choice fails in the FM model*. Indeed, let us assume that the axiom of choice is valid in the FM model. We choose arbitrarily two atoms from A. We denote them by a_1, a_2, and by pairing we obtain the set $\{a_1, a_2\}$. From the cofinite set $A \setminus \{a_1, a_2\}$ we take arbitrarily another two atoms, we denote them by a_3, a_4, and by pairing we obtain the set $\{a_3, a_4\}$. In this way we obtain an *infinite* family of disjoint sets denoted by $\{a_1, a_2\}$, $\{a_3, a_4\}$, ... (note that the indexing on the set A is only a writing convention; we do not say that A is countable). Now, by the axiom of choice, we should be able to find a set M which contains *exactly one* element from each of sets $\{a_1, a_2\}$, $\{a_3, a_4\}$, ... However $M \subset A$ should be infinite and coinfinite at the same time. This contradicts the structure of A.

The proof of the next result is simple, and uses only the definition of support [5].

Proposition 2.21.

1. *Let X and Y be nominal sets. For each $x \in X$ and $y \in Y$, we have $supp((x, y)) = supp(x) \cup supp(y)$.*
2. *Let X be a finite FM-set. Then $supp(X) = \cup\{supp(x) \mid x \in X\}$.*

Remark 2.22. *The following properties are consequences of the support definition:*

1. *If $a, b \in A$, then $(a \neq b \Longleftrightarrow a\#b)$.*
2. *If $a, b \in A$ and $a, b\#x$, then $(a\,b) \cdot x = x$.*
3. *Let M be the monoid generated by substitutions of form $\{b|a\}$. If $y\#M$ and $y\#x$ then $y\#Mx$.*

Proposition 2.23. *If $(x_i)_i$ is a set of distinct variables and p a formula in the logic of FM, then we have the following implications:*

$$[\forall a.(a\#(x_i)_i \Rightarrow p)] \Rightarrow [\text{И}a.p] \Rightarrow [\exists a \in A(a\#(x_i)_i \wedge p)].$$

Proof. We know that the set $\{a \in A \mid a\#(x_i)_i\}$ is cofinite. Assuming that $\forall a.(a\#(x_i)_i \Rightarrow p)$, we have $\{a \in A \mid a\#(x_i)_i\} \subset \{a \in A \mid p\}$ and so, $\text{И}a.p$. Now, if we assume $\text{И}a.p$, then $\{a \in A \mid p\}$ is cofinite, and so, $\{a \in A \mid a\#(x_i)_i\} \cap \{a \in A \mid p\}$ is the intersection of two cofinite subsets of the infinite set A which cannot be empty. Indeed, if we assume that the intersection $B \cap C$ of two cofinite subsets of A is empty, we have $C_{B \cap C} = A$, and so $C_B \cup C_C = A$; because C_B and C_C are finite, we obtain A is finite which contradicts axiom 10. □

Remark 2.24. *If the free variables of the formula p are contained in the set of distinct variables $\{a, (x_i)_i\}$, we also have the converse implication: $[\exists a \in A(a\#(x_i)_i \wedge p)]$ implies $[\forall a.(a\#(x_i)_i \Rightarrow p)]$. Indeed, let us suppose that for some $a \in A$ we have $a\#(x_i)_i \wedge p$. Then, by Remark 2.16, we obtain that for every atom b we have $p(b, (a\,b)\cdot(x_i)_i)$, where $p(b, (a\,b)\cdot(x_i)_i)$ denotes the formula obtained when we substitute b for all free occurrences of a in p, and $(a\,b) \cdot x_j$ for all free occurrences of x_j in p. Now, if we suppose $b\#(x_i)_i$, we get $(a\,b)\cdot(x_i)_i = (x_i)_i$ (because we also have $a\#(x_i)_i$), and so we obtain $p(b, (x_i)_i)$.*

Definition 2.25. Let X be a nominal set and $u \in X$. If $B \subset A$, we define $u||B \stackrel{def}{=} \{\pi \cdot u \,|\, \pi \in Fix(B)\}$; $u||B$ is called *the freshness orbit of u on B*.

Remark 2.26. *$u||B$ is an equivalence class of the sets which are equal up to a renaming of atoms which are not in B (because for $\pi \in Fix(B)$ we have $u||B = (\pi \cdot u)||B$). For example, we have $a||\{b\} = \{a, c, d, e, ...\}$.*

The abstraction function is used for binding operators in various calculi. By now on, we consider that the set A is equipped with an interchange function $\cdot : S_A \times X \to X$ defined by $\pi \cdot a = \pi(a)$ for all $\pi \in S_A$ and all $a \in A$. This means that by now on we consider A be an NFM-set.

Definition 2.27. Let X be a nominal set.

1. *For $a \in A$ and $x \in X$ we can define an* abstractive element *to be of form $[a]x$ where $[a]x = \cap\{V \subset A \times X \,|\, (a, x) \in V \wedge supp(V) \subset supp(x) \setminus \{a\}\}$.*
2. *We define the* abstraction function *to be the function $abs : A \times X \to [A]X = \{[a]x \,|\, a \in A \wedge x \in X\}$ defined by the correspondence $(a, x) \to [a]x$.*

We often use the notion of α-abstraction for abstractive elements by analogy with the abstraction in the λ-calculus. Corollary 2.29 and Example 2.14 make this analogy possible. We can now prove that $[a]x = (a, x)||(supp(x) \setminus \{a\})$.

Theorem 2.28. *Let X be a nominal set. If $a \in A$ and $x \in X$, then $[a]x = (a, x)||(supp(x) \setminus \{a\})$. This means that $[a]x$ is the freshness orbit of (a, x) on $supp(x) \setminus \{a\}$.*

Proof. Let $U = (a, x)||(supp(x) \setminus \{a\})$ and $\pi \in Fix(supp(x) \setminus \{a\})$. We have $\pi \cdot U = \{(\pi \cdot a^{'}, \pi \cdot x^{'}) \,|\, (a^{'}, x^{'}) \in U\}$. However $(a^{'}, x^{'}) \in U$ only when $a^{'} = \pi^{'} \cdot a$ and $x^{'} = \pi^{'} \cdot x$ with $\pi^{'} \in Fix(supp(x) \setminus \{a\})$. Since $Fix(supp(x) \setminus \{a\})$ is a group, we have $\pi \cdot U = U$ which means that $[a]x \subset U$. Conversely, we have $(a, x) \in [a]x$ and $\pi \cdot [a]x = [a]x$ for all $\pi \in Fix(supp(x) \setminus \{a\})$. Therefore, for all $\pi \in Fix(supp(x) \setminus \{a\})$, we have $(\pi(a), \pi \cdot x) \in [a]x$, and so $U \subset [a]x$. □

Corollary 2.29. *Let X be a nominal set, $a \in A$ and $x \in X$. Then we have*
a) $[a]x = \{(y, (y, a)x) \,|\, y \in A, y \neq a \text{ and } y\#x\} \cup \{(a, x)\}$;
b) $supp([a]x) = supp(x) \setminus \{a\}$.

Proof. The first part is obvious. For the second part we use Proposition 2.21 which says that $supp((x, y)) = supp(x) \cup supp(y)$ (the pair (x, y) is defined as in the ZF model, by $(x, y) = \{\{x\}, \{x, y\}\}$) for $x, y \in X$, and $supp((a, x)) = supp(x) \cup \{a\}$ for $a \in A$ and $x \in X$. If $B \subset A$ and $supp(u) \setminus B \neq \emptyset$, then by easy computation and using Example 2.19 and Definition 2.25, we obtain $supp(u||B) = B$.

Coming back to our proof, using Theorem 2.28 we get $supp([a]x) = supp((a, x)||(supp(x) \setminus \{a\})) = supp(x) \setminus \{a\}$ because $supp((a, x)) \setminus (supp(x) \setminus \{a\}) = \{a\} \neq \emptyset$. □

Example 2.30.
$[a](A\setminus\{a\}) = \{(a, A\setminus\{a\}), (b, A\setminus\{b\}), (c, A\setminus\{c\}), ...\}$
$[a]\{a, b\} = \{(a, \{a, b\}), (c, \{c, b\}), (d, \{d, b\}), (e, \{e, b\}), ...$
This means that b is free, and only a is bound (in the sense of Remark 2.33).

According to [5], we have the following result:

Proposition 2.31. *Let X be a nominal set, $a, b \in A$ and $x, y \in X$.*
We have $[a]x = [b]y$ if and only if one of the following statements is true:

- *$a = b$ and $x = y$.*
- *$a \neq b$, $b\#x$ and $y = (b\,a) \cdot x$.*

Corollary 2.32. *Let X be a nominal set, $a, b \in A$ and $x, y \in X$.*
If we have $c\#(a, b, x, y)$ and $[a]x = [b]y$, then $(a\,c) \cdot x = (b\,c) \cdot y$.

Remark 2.33. *For a λ-term x, we denote by $fn(x)$ the free names (variables) for x. According to Corollary 2.29 and Example 2.14, we can say that $fn([a]x) = fn(x) \setminus \{a\}$. This means that* [] can be seen as a binding operators of a in x. *For more details about free names and support, see [5] and [6]. However the bound names are not defined in the FM approach and so, we define the abstraction (function) to be similar to the usual (λ-calculus) binding. Also, it is important to remark that $[a]x$ is an equivalence class on renamings.*

In the previous results about abstraction we can consider X to be the nominal set $FM(A)$, and the elements of X to be the FM sets. Hence whenever x is an FM-set, the element $[a]x$ can be defined as in Definition 2.27; the element $[a]x$ is equal with $\{(y, (y, a)x) \mid y \in A, y \neq a \text{ and } y\#x\} \cup \{(a, x)\}$ which is an FM-set because of axiom 6 of the FM set theory. Whenever $a \in A$ and x is an FM-set, we have $supp([a]x) = supp(x) \setminus \{a\}$. Proposition 2.31 and Corollary 2.32 are valid when X is $FM(A)$ *i.e.* x, y are FM-sets.

3. Mobility in the π-calculus

The π-calculus [9] is a widely accepted as a model of interacting systems with dynamically evolving communication topology. There exist attempts to define abstract machines for the π-calculus [3]. The π-calculus works with process expressions, and allows channels to be passed along other channels. This feature leads to mobility, expressed by changing configurations and connectivity among processes. This mobility increases expressive power, enabling the description of many high-level concurrent features. π-calculus has several semantics. Among them, we mention both the unlabeled semantics, and early and the late labeled semantics [13]. The labeled semantics will be compared below with the nominal semantics.

The computational world of the π-calculus contains just processes (also called agents) and channels (also called names or ports). It can model networks in which messages are sent from one site to another site, and may contain links to active processes or to other sites. The π-calculus is a general model of computation which takes interaction as primitive. We present here the monadic version of the π-calculus: this means that a message consists of exactly one name. Let $\mathcal{X}$ be a infinite set of *names*. The elements of $\mathcal{X}$ are denoted by $x, y, z \ldots$ The terms of this formalism are called processes and processes are denoted by $P, Q, R \ldots$

Definition 3.1. The *processes* are defined over the set $\mathcal{X}$ of names by the grammar

$$P ::= 0 \mid \overline{x}\langle z\rangle.P \mid x(y).P \mid P \mid Q \mid P + Q \mid !P \mid new\, xP$$

The empty process is denoted by 0. The other process expressions are defined by guarded processes $\overline{x}\langle z\rangle.P$ and $x(y).P$, parallel composition $P \mid Q$ (that is the components P and Q can proceed independently and can interact via shared names), nondeterministic choice $P + Q$, replication $!P$ and a restriction $new\, xP$ creating a local fresh channel x for the process P (components of P can use x to interact with one another but not with other processes). π-calculus replication $!P$ can also be expressed by recursive equations of parametric processes ($!P$ can be thought as an infinite composition $P|P|...$). The guards are input guards and output guards. They represent sending and receiving a message (name) along a channel. The output guarded process $\overline{x}\langle z\rangle.P$ sends z along x and then, after the output has completed, continues as P. An input guarded process $x(y).Q$ waits until a name is received along x, substitutes it for the bound variable y and continues as Q. The parallel composition $\overline{x}\langle z\rangle.P \mid x(y).Q$ may synchronize two processes along a channel x. The processes can interact by using names they share. A name received in one interaction can be used in another; by receiving a name, a process can interact with processes which are unknown to it, but which now share the same channel name. π-calculus mobility stems from its scoping of names and extrusion of names from their scopes.

There is an important distinction between input and output guards. The output guard is a simple sending of a name z along a channel x, but the input guard has a more complex action: the name received along the channel x replaces y in the process following the input guard. Input guard is a *binding* operator involving substitutions. In $x(y).P$, the name y binds free occurrences of y in P. In a second binding operator $new\, xP$, the name x binds free occurrences of x in P. After this discussion we are able to give the following definition:

Definition 3.2. In each $x(z).P$ and $new\, zP$, the displayed occurrences of z is binding with scope P. An occurrence of a name in a process is *bound* if it is within the scope of a binding occurrence of a name. An occurrence of a name in a process is *free* if it is not bound.

We write $fn(P)$ for the set of names that have a free occurrence in P. It is straightforward to show by induction that $fn(P)$ is finite for every P. The free names of process circumscribe its capabilities for action: for a name x, in order for P to send x, to send via x or to receive via x, it must be that $x \in fn(P)$. Thus in order for two processes to interact via a name, that name must occur free in both of them, in one case expressing a capability to send, and in another a capability to receive.

For the binding of a name in a process $x(z).P$, the free occurrences of z in P indicate the places where the name received via x is substituted when the process acts. It is by means of such substitutions of names for names that change of connectivity among the components of a system is expressed. For the second binding operator $new\, zP$, it means that the components of P can use z to interact with one another but not with other processes. Also, in $new\, z\, P$, components of P can also pass z to one another, and they can extrude the scope of z by sending z via some other name.

Definition 3.3. (α-convertibility)

1. If the name y does not occur in the process P, then $P\{y|z\}$ is the process obtained by replacing each free occurrence of z in P by y.
2. A *change of bound names* in a process P is the replacement of a subterm $x(z).Q$ of P by $x(y).Q\{y|z\}$, or the replacement of a subterm $new\, zQ$ of P by $new\, yQ\{y|z\}$, where in each case y does not occur in Q.
3. The processes P and Q are *α-convertible (denoted by $P =_\alpha Q$ or just $P = Q$)* if Q can be obtained from P by a finite number of changes of bound names.

Further on, we make the following conventions:

1. Processes that are α-convertibility are identified.
2. When considering a collection of processes and substitutions, we assume that the bound names of the processes are chosen to be different from their free names and from the names of substitutions.

A structural congruence relation can be defined over the set of processes; this relation provides a static semantics of some formal constructions. To define a structural congruence, we need the notions of context and congruence. First, a minor auxiliary definition, needed because only summations can be operands in sums: an occurrence 0 in a process is *degenerate* if it is the left or right term in a sum $P + Q$, and *non-degenerate* otherwise.

Informally, a context differs from a process only in having the *hole* $[\cdot]$ in place of a non-degenerate occurrence of 0. Contexts that are α-convertible are not identified, however: a context is regarded as a syntactic entity that transforms processes into processes. Formally, we have:

Definition 3.4. (context). A *context* is obtained when the hole $[\cdot]$ replaces a non-degenerate occurrence of 0 in a process given by the grammar in Definition 3.1.

If C is a context and P a process, we write $C[P]$ for the process obtained by replacing the $[\cdot]$ in C by P. The replacement is literal, so names free in P may be bound in $C[P]$.

We are now able to give a formal definition for congruence:

Definition 3.5. (congruence) An equivalence relation $\Re$ on processes is a *congruence* if $(P, Q) \in \Re$ implies $(C[P], C[Q]) \in \Re$ for every context C.

The following set of rules gives the definition of structural congruence:

Definition 3.6. The relation $\equiv$ over the set of processes is called *structural congruence*, and is defined as the smallest congruence which satisfies

- $P \equiv Q$ if $P =_\alpha Q$ $\quad P + 0 \equiv P$, $P + Q \equiv Q + P$ $\quad (P + Q) + R \equiv P + (Q + R)$
- $P \mid 0 \equiv P$, $\;P \mid Q \equiv Q \mid P$, $\;(P \mid Q) \mid R \equiv P \mid (Q \mid R)$, $\quad !P \equiv P \mid !P$
- $new\, x0 \equiv 0$, $newx(newyP) \equiv new\, y(new\, xP)$,
  $new\, x(P \mid Q) \equiv P \mid new\, xQ$ if $x \notin fn(P)$.

Structural congruence deals with the aspects related to the structure of the processes. The axiom $new\, x(P \mid Q) \equiv P \mid new\, xQ$ if $x \notin fn(P)$ expresses that a restriction can be moved so as to include in or exclude from its scope a process in which the restriction name is not free. It clearly shows that *new* is a static binder (in the last part of this paper we shall see *new* as the binding in nominal logic and we shall express the nominal semantics of the π- calculus)

The dynamical evolution of a process is described in π-calculus by a reaction relation over processes. The reaction relation contains those transitions which can be inferred from a set of rules.

Definition 3.7. The simple operational semantics of the π-calculus is defined as the smallest relation $\rightarrow$ satisfying the following rules.

(com) $\quad (\overline{x}\langle z\rangle.P + R_1) \mid (\, x(y).Q + R_2) \rightarrow P \mid Q\{z/y\}$

(par) $\quad P \rightarrow Q$ implies $P \mid R \rightarrow Q \mid R$

(res) $\quad P \rightarrow Q$ implies $new\, xP \rightarrow new\, xQ$

(str) $\quad P \equiv P',\ P' \rightarrow Q'$ and $Q' \equiv Q$ implies $P \rightarrow Q$

A simple reaction is defined by these rules.

It is clear that a reaction is closed under $\equiv$ since $P \rightarrow Q$ and $R \equiv Q$ implies $P \rightarrow R$.

3.1. Example of π-calculus Mobility

The π-calculus is able to describe mobile systems, providing a conceptual framework and mathematical tools. The word mobility is used with many meanings. The π-calculus deals with the mobility given by links that move in a space of linked processes. For example, hypertext links can be created, passed around and can disappear; or references can be passed as arguments of remote method invocations in object-oriented systems. Our example describes a simple interaction between a handphone carried in a car and some base stations. The system is described in Figure 1. The connections between the car (handphone) and the base stations can change as the car is moving around.

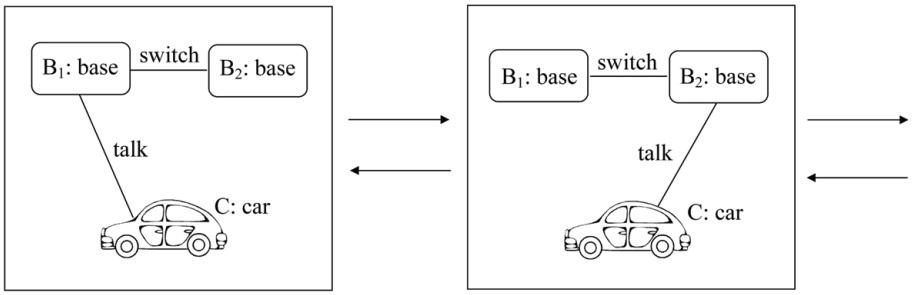


Fig. 1. Mobility of links between a car and two stations.

We consider three processes B_1, B_2 and C corresponding to two bases and the car respectively. We start with their parallel composition $B_1 \mid C \mid B_2$ described by the left picture of Figure 1. The base B_1 and car C are connected by a channel *talk* and B_1

and B_2 by a channel *switch*. This means that *talk* is free in both B_1 and C, and *switch* is a free name in both B_1 and B_2. By the process expression $new\ talk(B_1 \mid C) \mid B_2$, the name *talk* is restricted to B_1 and C, and we interpret that B_1 and C have exclusive communication along the channel *talk*. If $B_1 = \overline{switch}\langle talk\rangle.B_1'$, then base B_1 wishes to send the name of channel *talk* to base B_2 along the channel *switch*. Moreover, if *talk* is not free in B_1' ($talk \notin fn(B_1')$, then B_1' loses its link to C. Base B_2 is waiting for a channel name sent by B_1, namely $B_2 = switch(y).B_2'$. Applying the corresponding reaction rules, we have the transition

$$new\ talk\ (B_1 \mid C) \mid B_2 \longrightarrow B_1' \mid new\ talk\ (C \mid B_2'')$$

where $B_2'' = B_2'\{talk/y\}$. The initial process $new\ talk\ (B_1 \mid C) \mid B_2$ changes its communication topology as shown in the right hand picture in Figure 1. Now B_2'' and C have exclusive communication along the channel *talk*. This is essentially the mobility mechanism offered by the π-calculus. More details are presented in [9].

3.2. Early Semantics of the π-calculus

Activity within a system is described by simple reactions. Simple reactions do not explain how a system can interact with its environment, however. In order to understand the behavior of a system by analyzing its parts, it is necessary to talk about the actions that the parts can perform. We shall define a family o labeled transition relations on processes. The transition relations are defined by inference rules. This section presents the rules for early semantics of the π-calculus. The transition rules are labeled by the actions. We have four kind of actions:

Definition 3.8. The *actions* are given by $\alpha ::= \overline{x}\langle y\rangle \mid xy \mid \overline{x}(z) \mid \tau$.

$\overline{x}\langle y\rangle$ is sending the name y via the name x, xy is receiving y via x and $\overline{x}(z)$ is sending a fresh name via x. τ is the internal action.

The transition relation labeled by α is written $\xrightarrow[e]{\alpha}$, where α symbolizes the action and e is a notation for the semantics of the π-calculus where we define the transition rule (in this section we work with the *early* semantics of the π-calculus). For example $P \xrightarrow[e]{\overline{x}\langle y\rangle} Q$ means that P can send y via x and evolve to Q; $P \xrightarrow[e]{xy} Q$ means that P can receive y via x and evolve to Q; $P \xrightarrow[e]{\overline{x}(y)} Q$ means that P can send a fresh name y via x and evolve to Q.

The free and bound names for each action are presented in the following table:

Action	Kind	fn	bn	$\alpha\sigma$
$\overline{x}\langle y\rangle$	free output	$\{x, y\}$	$\emptyset$	$\overline{x\sigma}\langle y\sigma\rangle$
xy	input	$\{x, y\}$	$\emptyset$	$x\sigma y\sigma$
$\overline{x}(z)$	bound output	$\{x\}$	$\{z\}$	$\overline{x\sigma}(z)$
τ	internal	$\emptyset$	$\emptyset$	τ

Definition 3.9. The *transition relations* $\{\xrightarrow[e]{\alpha} \mid \alpha \text{ is an action}\}$ are defined by the rules of Table 1.

Table 1. Early Semantics of the π-calculus

$$OUT: \frac{}{\overline{x}\langle y\rangle.P \xrightarrow[e]{\overline{x}\langle y\rangle} P} \quad INP: \frac{}{x(z).P \xrightarrow[e]{xy} P\{y|z\}} \quad PAR_L: \frac{P \xrightarrow[e]{\alpha} P',\ bn(\alpha)\cap fn(Q)=\emptyset}{P \mid Q \xrightarrow[e]{\alpha} P' \mid Q}$$

$$SUM_L: \frac{P \xrightarrow[e]{\alpha} P'}{P+Q \xrightarrow[e]{\alpha} P'} \quad COM_L: \frac{P \xrightarrow[e]{\overline{x}\langle y\rangle} P', Q \xrightarrow[e]{xy} Q'}{P \mid Q \xrightarrow[e]{\tau} P' \mid Q'} \quad OPEN: \frac{P \xrightarrow[e]{\overline{x}\langle z\rangle} P',\ z \neq x}{new\, zP \xrightarrow[e]{\overline{x}(z)} P'}$$

$$TAU: \frac{}{\tau.P \xrightarrow[e]{\tau} P} \quad CLOSE_L: \frac{P \xrightarrow[e]{\overline{x}(z)} P', Q \xrightarrow[e]{xz} Q'}{P \mid Q \xrightarrow[e]{\tau} new\, z(P' \mid Q')}, \text{ when } z \notin fn(Q)$$

$$RES: \frac{P \xrightarrow[e]{\alpha} P'}{new\, zP \xrightarrow[e]{\alpha} new\, zP'}, z \notin n(\alpha) \quad REP_{com}: \frac{P \xrightarrow[e]{\overline{x}\langle y\rangle} P', P \xrightarrow[e]{xy} P''}{!P \xrightarrow[e]{\tau} (P' \mid P'') \mid !P}$$

$$REP_{act}: \frac{P \xrightarrow[e]{\alpha} P'}{!P \xrightarrow[e]{\alpha} P' \mid !P} REP_{close}: \frac{P \xrightarrow[e]{\overline{x}(z)} P', P \xrightarrow[e]{xz} P''}{!P \xrightarrow[e]{\tau} (new\, z(P' \mid P'')) \mid !P} \text{ when } z \notin fn(P)$$

Note especially that $x(z).P$ can receive *any* name via x, and when a name is received, it is substituted for the placeholder z in P. The following result, called the Harmony Lemma, is known from [13].

Theorem 3.10. *We have the following connections between the simple reactions and the transition relations:*

1. $P \equiv \xrightarrow[e]{\alpha} P'$ *implies* $P \xrightarrow[e]{\alpha} \equiv P'$.
2. $P \rightarrow P'$ *if and only if* $P \xrightarrow[e]{\tau} \equiv P'$.

3.3. Late Semantics of the π-calculus

The transition $P \xrightarrow[e]{xy} Q$ expresses that P can receive the name y via x and evolve to Q. An action of the form xy is sometimes referred to as a free input; it records both the name used for receiving and the name received. In the late relations, the free-input actions are replaced by the input prefixes, referred to in this context as *bound-input actions*. In late transitions $P \xrightarrow[l]{x(z)} Q$, the label contains a placeholder z for the name to be received, rather than the name itself. If $\alpha = x(z)$ the $fn(\alpha) = x$ and $bn(\alpha) = z$. We denote a late transition relation $P \xrightarrow{\alpha} Q$ by $P \xrightarrow[l]{\alpha} Q$, where α symbolizes the action and l is a notation for the semantics of the $\pi - calculus$ where we define the transition rule (in this section we work with the *late* semantics of the $\pi - calculus$).

The late semantics is given by the transition rules presented in Table 2. Usually in labeled transition systems for π-calculus we use the following actions: $\overline{x}\langle y\rangle \mid x(y) \mid \overline{x}(z) \mid \tau$. The free and bound names for each action are presented in the following table:

Action	Kind	fn	bn	$\alpha\sigma$
$\overline{x}\langle y\rangle$	free output	$\{x, y\}$	$\emptyset$	$\overline{x\sigma}\langle y\sigma\rangle$
$x(y)$	input	$\{x\}$	$\{y\}$	$x\sigma y\sigma$
$\overline{x}(z)$	bound output	$\{x\}$	$\{z\}$	$\overline{x\sigma}(z)$
τ	internal	$\emptyset$	$\emptyset$	τ

Table 2. Late Semantics of the π-calculus

$$L\text{-}OUT: \frac{}{\overline{x}\langle y\rangle.P \xrightarrow[l]{\overline{x}\langle y\rangle} P} \qquad L\text{-}INP: \frac{}{x(z).P \xrightarrow[l]{x(z)} P}$$

$$L\text{-}PAR_L: \frac{P \xrightarrow[l]{\alpha} P'}{P \mid Q \xrightarrow[l]{\alpha} P' \mid Q}, \quad bn(\alpha) \cap fn(Q) = \emptyset \qquad L\text{-}SUM_L: \frac{P \xrightarrow[l]{\alpha} P'}{P + Q \xrightarrow[l]{\alpha} P'}$$

$$L\text{-}COM_L: \frac{P \xrightarrow[l]{\overline{x}\langle y\rangle} P', Q \xrightarrow[l]{x(z)} Q'}{P \mid Q \xrightarrow[l]{\tau} P' \mid Q'\{y/z\}} \qquad L\text{-}OPEN: \frac{P \xrightarrow[l]{\overline{x}\langle z\rangle} P'}{new\, zP \xrightarrow[l]{\overline{x}(z)} P'}, \quad z \neq x$$

$$L\text{-}TAU: \frac{}{\tau.P \xrightarrow[l]{\tau} P} \qquad L\text{-}CLOSE_L: \frac{P \xrightarrow[l]{x(z)} P', Q \xrightarrow[l]{\overline{x}(z)} Q'}{P \mid Q \xrightarrow[l]{\tau} new\, z(P' \mid Q')}$$

$$L\text{-}RES: \frac{P \xrightarrow[l]{\alpha} P'}{new\, zP \xrightarrow[l]{\alpha} new\, zP'}, z \notin n(\alpha) \qquad L\text{-}REP_{act}: \frac{P \xrightarrow[l]{\alpha} P'}{!P \xrightarrow[l]{\alpha} P' \mid !P}$$

$$L\text{-}REP_{com}: \frac{P \xrightarrow[l]{\overline{x}\langle y\rangle} P', P \xrightarrow[l]{x(z)} P''}{!P \xrightarrow[l]{\tau} (P' \mid P''\{y/z\}) \mid !P} \qquad L\text{-}REP_{close}: \frac{P \xrightarrow[l]{\overline{x}(z)} P', P \xrightarrow[l]{x(z)} P''}{!P \xrightarrow[l]{\tau} (new\, z(P' \mid P'')) \mid !P}$$

In late semantics we use a transition $x(z).P \xrightarrow[l]{x(z)} P$ where the label $x(z)$ indicates a placeholder z for the name to be received along channel x (rather than the name itself). This means that we replace the *free-input* actions with *bound-input* actions. Using the late rules, the placeholder z is instantiated by y when the *communication* is inferred, rather than when the *input* by the receiver is inferred. Thus, the early-late terminology is based on when a name is instantiated in inferring an interaction: when the input is inferred (early), or when the interaction is inferred (late).

We recall that we denote the late relations by $\xrightarrow[l]{}$ and the early relations by $\xrightarrow[e]{}$. By induction on inference depth, we can prove that there is a close relationship between the early and late semantics.

Theorem 3.11.([13])

1. $P \xrightarrow[e]{\overline{x}\langle z\rangle} P'$ *if and only if* $P \xrightarrow[l]{\overline{x}\langle z\rangle} P'$.
2. $P \xrightarrow[e]{xz} P'$ *if and only if there are* Q *and* y *such that* $P \xrightarrow[l]{x(y)} Q$ *and* $P' = Q\{z|y\}$.
3. $P \xrightarrow[e]{\overline{x}(z)} P'$ *if and only if* $P \xrightarrow[l]{\overline{x}(z)} P'$.

4. $P \xrightarrow[e]{\tau} P'$ *if and only if* $P \xrightarrow[l]{\tau} P'$.

More details on the late semantics of the π-calculus are in [13].

4. Nominal Semantics of the π-calculus

We try now to analyze how we can use the FM framework. Using the same arguments as in Example 2.14, we can say that the set of the π-calculus terms form an NFM-set, and the set of the π-calculus terms modulo α-conversion form also an NFM-set and an FM-set. If we organize the set of variables as an NFM-set (variables in π-calculus are atoms, and the set A of atoms can be organized as an NFM-set with the S_A-action given in Example 2.7), we can define the abstraction between a variable and a π-calculus term in the sense of Definition 2.27. In π-calculus we have two binding operators; in each of the expressions $x(y).P$ and $new\, yP$, we have y bound. We can use the notion of abstraction, and write these expressions like $x[y]P$ and $new\,[y]P$. In this case we can see bindings represented by *new* and *input* as α-abstractions (or abstractive elements of Definition 2.27) defined in FM. However, we do not unify *new* and *input* by the same α-abstraction.

Remark 4.1. *We recall that in the FM approach there are no bound names. The "binding" made by new and input can be seen as α-abstraction in the sense of Definition 2.27. However the notion of α-abstraction is used to emphasize that, when we apply new zP or $x(z).P$, we eliminate z from $supp(P)$ which intuitively is a "binding of $supp(z)$ in P" (we know that $supp([z]P) = supp(P) - \{z\}$ and the support of a process P is precisely the set of free names of P). From now on, we shall simply call these α-abstractions generated by new and input as* ***bindings****.*

Using Proposition 2.31 we obtain a property of the binding operators in the π-calculus.

Proposition 4.2.

1. *If $a[x]P = b[y]Q$, then $a = b$ and one of the following sentences is true:*
   - *$x = y$ and $P = Q$.*
   - *$x \neq y$ and $x\#Q$ and $P = (x\,y) \cdot Q$.*
2. *If $new[x]P = new[y]Q$, then one of the following sentences is true:*
   - *$x = y$ and $P = Q$.*
   - *$x \neq y$ and $x\#Q$ and $P = (x\,y) \cdot Q$.*

Definition 4.3. If σ is a substitution of the form $\{z|y\}$ we define $P\sigma$ by giving the rules:

1. $0\sigma = 0$
2. $(P + Q)\sigma = P\sigma + Q\sigma$
3. $(P|Q)\sigma = (P\sigma)|(Q\sigma)$
4. $(\pi.P)\sigma = \pi\sigma.P\sigma$
5. $(new\, aP)\sigma = new\, a(P\sigma)$
6. $(!P)\sigma =!(P\sigma)$

In fact $P\sigma$ is the process P whose names have been replaced according to substitution σ, in a capture-free way. When we consider some processes and substitutions, we assume that the bound names of the processes are chosen to be different from their free names and from the names of substitutions. However this assumption is possible

because of the finite support property. In the nominal framework we can formalize this convention of choosing fresh names using the nominal quantifier (Definition 4.4). In the nominal approach the actions of substitutions on processes are defined by the rules in the following Definition 4.4, in which M is the monoid generated by the substitutions of the form $\{b|a\}$.

Definition 4.4. The actions of the monoid of substitutions M on processes P, Q are:

1. $0M = 0$ 2. $(P+Q)M = PM + QM$ 3. $(P|Q)M = (PM)|(QM)$
4. $(!P) = !PM$ 5. $(\overline{x}\langle y\rangle.P)M = \overline{xM}\langle yM\rangle.PM$
6. $\forall x.И a.\forall P.(x[a]P)M = xM[a]PM$ (action for bound input)
7. $И a.\forall P.\ (new[a]P)M = new\,[a]PM$ (action for new)

In fact this definition is a natural generalization of Definition 4.3 for more complex finite substitutions of the form $\{a_1, a_2, ...|b_1 b_2...\}$.. The freshness conditions assumed (by convention) after Definition 4.3 are encoded using И.

In the nominal approach we do not define a structural congruence; however the processes which are obtained from one another by an α-conversion (i.e. a changing of bound names) are identified in the nominal framework (Example 2.14).

In the new nominal semantics of the π-calculus, in a labeled transition $P \xrightarrow[n]{\alpha} Q$ we assume (by convention) that α has no bound names (i.e. α is not an input nor a bound output). In a labeled transition $P \xrightarrow[n]{x\,bn\,\alpha} Q$, $x\,bn\,\alpha$ means that x is a fresh name sent by α. Note that $x\,bn\,\alpha$ is just a notation which allow us to emphasize the fresh name x in the action α. In fact $z\,bn\,\bar{x}\langle z\rangle$ is exactly the bound output action $\bar{x}(z)$. Precisely the notation $z\,bn\,\bar{x}\langle z\rangle$ represents "sending the fresh name z via x". To avoid any possible confusion we make the remark that $x\,bn\,\alpha$ does not represent the binding of x only in the action α, and not in the derivation; $x\,bn\,\alpha$ is just another notation for the bound output action ($z\,bn\,\bar{x}\langle z\rangle$ is just another notation for $\bar{x}(z)$, and hence $z\,bn\,\bar{x}\langle z\rangle$ and $\bar{x}(z)$ have the same meaning). The symbol bn is used only to suggest that x is bound by new in the agent.

Definition 4.5. The labeled transition rules of Figure 2 define the *nominal semantics* of the π-calculus. A transition of the form $P \xrightarrow{u} Q$ (where u is the action) in the nominal semantics of the π-calculus is denoted by $P \xrightarrow[n]{u} Q$.

Remark 4.6. *In the transition rules of Figure 2 and in the rules of Definition 4.4 the bindings made by new and input (in the sense of Remark 4.1) are seen as local bindings only for their process (for example in the expressions $x[y]P$ and $new[y]P$, the free occurrences of y are bound only in P), and the bindings made by the quantifiers $\forall$ and И are seen as global bindings for the entire rule. So, it is possible to have, in some rules, expressions of forms:*

$$\forall y, ...formula_1(x[y]P, ...) \qquad \forall y, ...formula_2(new[y]P, ...)$$
$$И y, ...formula_3(x[y]P, ...) \qquad И y, ...formula_4(new[y]P, ...).$$

This does not mean that y is bound twice in the same expression. We make the distinction between local binding (which is the binding only for the indicated process) and global binding (which is the binding for the entire expression). Consequently, $\forall y, \ldots$

$formula_1(x[y]P,...)$ *and* $\forall y,....formula_2(new[y]P,...)$ *mean that* $formula_1(x[y]P,...)$ *and respectively* $formula_2(new[y]P,...)$ *are valid and* their validity does not depend on y. *Also,* $И y,....formula_3(x[y]P,...)$ *and* $И y,....formula_4(new[y]P,...)$ *mean that* $formula_3(x[y]P,...)$ *and respectively* $formula_4(new[y]P,...)$ *are valid for all but finitely many* y *(eventually these are valid iff* y *is fresh for an expression or for a process).*

For example, the expression $\forall x,y.И z.\forall P,Q.\dfrac{P \xrightarrow[n]{x(y)} Q}{new[z]P \xrightarrow[n]{x(y)} new[z]Q}$ *is correctly written even if we might be tempted to say that* y *and* z *are bound twice. The variables* x, y *and* z *are globally bound once. The second binding, made by new and input, is a local one and it is restricted to the processes* P *and* Q. *So, to say that* $\forall x,y.И z.\forall P,Q.\dfrac{P \xrightarrow[n]{x(y)} Q}{new[z]P \xrightarrow[n]{x(y)} new[z]Q}$, *is the same as saying: "*$P \xrightarrow[n]{x(y)} Q$ *implies* $new[z]P \xrightarrow[n]{x(y)} new[z]Q$*" if* z *satisfy some cofiniteness conditions (we shall prove later that these conditions are* z *is fresh both for* x *and for* y*).*

Fig. 2. Nominal Semantics of the π-calculus.

1. $\forall x,y,P.x[y]P \xrightarrow[n]{x(y)} P$ *(bound input)*

2. $\forall P,R,\alpha,Q.\dfrac{P \xrightarrow[n]{\alpha} Q}{P|R \xrightarrow[n]{\alpha} Q|R}$ *(free composition)*

3. *For bound concurrent composition we have the following two labeled transition rules (for bound output, respectively bound input):*

(a) $\forall R.Иx.\forall \alpha,P,Q.\dfrac{P \xrightarrow[n]{x\ bn\ \alpha} Q}{P|R \xrightarrow[n]{x\ bn\ \alpha} Q|R}$ (b) $\forall R.И y.\forall x,P,Q.\dfrac{P \xrightarrow[n]{x(y)} Q}{P|R \xrightarrow[n]{x(y)} Q|R}$

4. $\forall \alpha.Иx.\forall P,Q.\dfrac{P \xrightarrow[n]{\alpha} Q}{new[x]P \xrightarrow[n]{\alpha} new[x]Q}$ *(free restriction)*

5. *For bound restriction we have the following two labeled transition rules (for bound output, respectively bound input):*

(a) $\forall x,\alpha.И y.\forall P,Q.\dfrac{P \xrightarrow[n]{x\ bn\ \alpha} Q}{new[y]P \xrightarrow[n]{x\ bn\ \alpha} new[y]Q}$

(b) $\forall x,y.И z.\forall P,Q.\dfrac{P \xrightarrow[n]{x(y)} Q}{new[z]P \xrightarrow[n]{x(y)} new[z]Q}$

6. $\forall P.P \xrightarrow[n]{\tau} P$ *(free tau)* 7. $\forall x,y,P.\overline{x}\langle y\rangle.P \xrightarrow[n]{\overline{x}\langle y\rangle} P$ *(free output)*

8. $\forall x.И y.\forall P,Q.\dfrac{P \xrightarrow[n]{\bar{x}\langle y\rangle} Q}{new[y]P \xrightarrow[n]{y\, bn\, \bar{x}\langle y\rangle} Q}$ *(bound output)*

9. $\forall \alpha,P,Q,R.\dfrac{P \xrightarrow[n]{\alpha} Q}{P+R \xrightarrow[n]{\alpha} Q}$ *(sum)*

10. $\forall P,Q,x,y,z,R,S.\dfrac{P \xrightarrow[n]{x(y)} R \,and\, Q \xrightarrow[n]{\bar{x}\langle z\rangle} S}{P|Q \xrightarrow[n]{\tau} (R\{z|y\})|S}$ *(free $\beta-rule$)*

11. $\forall P.И z.\forall Q,R,S,x,y.\dfrac{P \xrightarrow[n]{x(y)} R \,and\, Q \xrightarrow[n]{z\, bn\, \bar{x}\langle z\rangle} S}{P|Q \xrightarrow[n]{\tau} new[z]((R\{z|y\})|S)}$ *(the mobility rule)*

12. $\forall P,\alpha,Q.\dfrac{P \xrightarrow[n]{\alpha} Q}{!P \xrightarrow[n]{\alpha} Q|!P}$ *(replication action)*

13. $\forall P,Q,x,y,z,R.\dfrac{P \xrightarrow[n]{x(y)} Q \,and\, P \xrightarrow[n]{\bar{x}\langle z\rangle} R}{!P \xrightarrow[n]{\tau} ((Q\{z|y\})|R)|!P}$ *(free replication)*

14. $\forall P.И z.\forall Q,R,x,y.\dfrac{P \xrightarrow[n]{x(y)} Q \,and\, P \xrightarrow[n]{z\, bn\, \bar{x}\langle z\rangle} R}{!P \xrightarrow[n]{\tau} new[z]((Q\{z|y\})|R)|!P}$ *(bound replication)*

Using these new transition rules we are able to prove that *new* (which generates new names) and *input* (which is not a fresh operator like, but is represented by substitution) have similar behaviors.

Let $(x_i)_i$ be a set of variables and p a formula. Proposition 2.23 and Remark 2.24 show us that, if the free variables of p are contained in the set $\{a,(x_i)_i\}$, then we obtain the following equivalence: $[\forall a.(a\#(x_i)_i \Rightarrow p)]$ if and only if $[И a.p]$. Now, we make a comparison between the known semantics of the π-calculus. First we can prove that all the rules 1-14 in the nominal semantics of the π-calculus, except two of them, are similar with those in the late semantics of the π-calculus. The exceptions are rule number 11 in the nominal semantics of the π-calculus and the rule for replication, respectively rule number 14. After presenting several nominal properties of transitions (Section 5) we are able to prove that even if these two rules are similar to some rules in the late semantics of the π-calculus. We start our comparison of semantics by defining a set of nominal transition rules as in the late semantics of the π-calculus (assuming some freshness conditions in the rule's hypothesis). Next we prove that these rules are in fact identical with those in the nominal semantics of the π-calculus.

Definition 4.7. A basic transition rule in the nominal semantics of π-calculus is in a *late-nominal form* if it coincides with one of the syntactic rules presented in Table 3.

Remark 4.8. *We know that a function such as bn is not defined in the nominal logic. In Table 3 $bn(\alpha)$ is only a notation for the name of α which is bound by new or*

input (the convention of using "bound" in nominal logic is expressed in Remark 4.1). A solution to eliminate bn from Table 3 is to split the transition rules which have conditions where bn is present into two rules-one for transitions with bound names and one for ones without. In our case, not to write so much, we present bn only as a notation *in certain rules, without a special meaning (bn in our case is not the function bn defined in the standard λ-calculus). If α is a bound input or a bound output where the name x is bound then x is* denoted *by bn(α) in that rule and if α has no bound names then ∅ is* denoted *by bn(α) in that rule.*

Now we present some results which allow us to say that *each transition rule in the nominal semantics of the π-calculus can be expressed in a late-nominal form.* The convention of notation presented in Remark 4.8 is often used in the proof of the following lemmas.

Lemma 4.9. *Using possibly different notations we have the following relationship between several transition rules in nominal semantics of the π-calculus and in Table 3:*

1. *Rule L-OUT$_{ln}$ in Table* 3 *is identical with rule number 7 in the nominal semantics of the π-calculus.*
2. *Rule L-INP$_{ln}$ in Table 3 is identical with rule number 1 in the nominal semantics of the π-calculus.*
3. *Rule L-TAU$_{ln}$ in Table 3 is identical with rule number 6 in the nominal semantics of the π-calculus.*
4. *Rule L-SUM$_{L_{ln}}$ in Table 3 is identical with rule number 9 in the nominal semantics of the π-calculus.*
5. *Rule L-COM$_{L_{ln}}$ in Table 3 is identical with rule number 10 in the nominal semantics of the π-calculus.*
6. *Rule L-REP$_{act_{ln}}$ in Table 3 is identical with rule number 12 in the nominal semantics of the π-calculus.*
7. *Rule L-REP$_{com_{ln}}$ in Table 3 is identical with rule number 13 in the nominal semantics of the π-calculus.*

Table 3. Late-Nominal Form of the Nominal Semantics of the π-calculus

$$L\text{-}OUT_{ln}: \frac{}{\overline{x}\langle y\rangle.P \xrightarrow[n]{\overline{x}\langle y\rangle} P} \qquad L\text{-}INP_{ln}: \frac{}{x[z]P \xrightarrow[n]{x(z)} P}$$

$$L\text{-}TAU_{ln}: \frac{}{\tau.P \xrightarrow[n]{\tau} P} \qquad L\text{-}COM_{L_{ln}}: \frac{P \xrightarrow[n]{\overline{x}\langle y\rangle} P', Q \xrightarrow[n]{x(z)} Q'}{P \mid Q \xrightarrow[n]{\tau} P' \mid Q'\{y/z\}}$$

$$L\text{-}SUM_{L_{ln}}: \frac{P \xrightarrow[n]{\alpha} P'}{P + Q \xrightarrow[n]{\alpha} P'} \qquad L\text{-}OPEN_{ln}: \frac{P \xrightarrow[n]{\overline{x}\langle z\rangle} P'}{new[z]P \xrightarrow[n]{\overline{x}(z)} P'}, \quad z\#x$$

$$L\text{-}PAR_{L_{ln}}: \frac{P \xrightarrow[n]{\alpha} P'}{P \mid Q \xrightarrow[n]{\alpha} P' \mid Q}, \quad bn(\alpha) \cap supp(Q) = \emptyset$$

(*bn* is only a convention of notation in the sense of Remark 4.8)

$$\textit{L-CLOSE}_{L_{ln}}: \quad \frac{P \xrightarrow[n]{x(y)} P', Q \xrightarrow[n]{\overline{x}(z)} Q'}{P \mid Q \xrightarrow[n]{\tau} new[z](P'\{z|y\} \mid Q')},\ z\#P$$

$$\textit{L-RES}_{ln}: \quad \frac{P \xrightarrow[n]{\alpha} P'}{new[z]P \xrightarrow[n]{\alpha} new[z]P'}\ when\ z \notin bn(\alpha)\ and\ z\#\alpha$$

(bn is only a convention of notation in the sense of Remark 4.8)

$$\textit{L-REP}_{act_{ln}}: \frac{P \xrightarrow[n]{\alpha} P'}{!P \xrightarrow[n]{\alpha} P' \mid !P} \qquad \textit{L-REP}_{com_{ln}}: \frac{P \xrightarrow[n]{\overline{x}\langle y\rangle} P', P \xrightarrow[n]{x(z)} P''}{!P \xrightarrow[n]{\tau} (P' \mid P''\{y/z\}) \mid !P}$$

$$\textit{L-REP}_{close_{ln}}: \quad \frac{P \xrightarrow[n]{x(y)} P', P \xrightarrow[n]{\overline{x}(z)} P''}{!P \xrightarrow[n]{\tau} (new[z](P'\{z|y\} \mid P'')) \mid !P}\ ,\ z\#P$$

Lemma 4.10. *Rule L-OPEN$_{ln}$ in Table 3 is identical with rule number 8 in the nominal semantics of the π-calculus.*

Proof. Let p be the formula: "$\forall P, P'.(P \xrightarrow[n]{\bar{x}\langle z\rangle} P'$ implies $new[z]P \xrightarrow[n]{z\ bn\ \bar{x}\langle z\rangle} P')$". The set of free variables of p is formed only by x. Indeed, in the formula p we have that P and P' are bound by the quantifier $\forall$ and z is bound by the binding operator new. The set of free variables of p is, then, contained in the set $\{z, x\}$ and so we have, for an arbitrary x, the equivalence result: "$\forall z.(z\#x$ implies $p)$" if and only if "$И z.p$". If we assume the free variables of p to be only the variables of p unbound globally, then these free variables of p would be z and x, and again these would be contained in the set $\{z, x\}$; hence, because of Proposition 2.23 and Remark 2.24 we have "$\forall z.(z\#x$ implies $p)$" if and only if "$И z.p$". However, $z\#x$ means $z \neq x$ because the support of the channel name x is formed only by x. Because the previous equivalence is valid for an arbitrary x we have that the rule *L-OPEN$_{ln}$* in Table 3 is identical with rule number 8 in the nominal semantics of the π-calculus. □

Lemma 4.11. *Rule L-PAR$_{L_{ln}}$ in Table 3 can be identified with one of the rules 3 (point (a) or (b)) or 2 in the nominal semantics of the π-calculus. This identification depends on the presence of the bound names in the action from the transition rule.*

Proof. Let p be the formula "$\forall P, P'.(P \xrightarrow[n]{\alpha} P'$ implies $P|Q \xrightarrow[n]{\alpha} P'|Q)$" where α is an action. If α has no bound names then we have that $bn(\alpha) = \varnothing$ (bn represents only a notation and not the function bn in λ-calculus; see Remark 4.8), p is always valid, and then the rule *L-PAR$_{L_{ln}}$* in Table 3 is identical with the rule number 2 in the nominal semantics of the π-calculus (we assumed that in the set of rules 1-14 in the nominal semantics of the π-calculus, a transition of the form $P \xrightarrow[n]{\alpha} P'$ means a transition without bound names). Now, let us assume that the action has a bound name. For example let x be the bound name of the action β. We chose an arbitrary Q. We study two cases: x can be bound by *new* or by *input*. Let q be the formula "$\forall \beta, P, P'.(P \xrightarrow[n]{x\ bn\ \beta} P'$ implies $P|Q \xrightarrow[n]{x\ bn\ \beta} P'|Q)$" ($x$ is bound by *new*) and r be the formula "$\forall y, P, P'.(P \xrightarrow[n]{y(x)} P'$ implies $P|Q \xrightarrow[n]{y(x)} P'|Q)$" (where x in bound by

input). The free variables of q are contained in the set $\{x, Q\}$. Indeed, the set of free variables of q is formed only by Q (precisely only by the set of free names of Q, which is included in Q) because P, P', β are bound by the quantifier $\forall$ and x is bound by *new*; if we assume the free variables of q to be only the variables of q unbound globally, then these free variables of q would be x,Q, and again these are contained in the set $\{x, Q\}$. The free variables of r are contained in the set $\{x, Q\}$. Indeed, the set of free variables of r is formed only by Q (more precisely, only by the set of free names of Q, which is included in Q) because P, P', y are bound by the quantifier $\forall$ and x is bound by *input*; if we assume the free variables of r to be only the variables of r unbound globally, then these free variables of r would be x,Q, and again these are contained in the set $\{x, Q\}$) Now, because of Proposition 2.23 and Remark 2.24, we obtain the equivalence results "$\forall x.(x\#Q$ implies $q)$" if and only if "$И x.q$", and respectively "$\forall x.(x\#Q$ implies $r)$" if and only if "$И x.r$". Because Q was chosen arbitrarily we can say that the rule $L\text{-}PAR_{L_{ln}}$ in Table 3 is exactly rule number 3 in the nominal semantics of the π-calculus (point (a) or (b) as x is bound by *new* respectively by *input*). □

Lemma 4.12. *Rule $L\text{-}RES_{ln}$ in Table 3 can be identified (by convention) with one of the rules 5 (point (a) or (b)) or 4 in the nominal semantics of the π-calculus. This identification depends on the presence of the bound names in the action from the transition rule.*

Proof. Now, let p be the formula "$\forall P, P'.(P \xrightarrow[n]{\alpha} P'$ implies $new[z]P \xrightarrow[n]{\alpha} new[z]P')$" where α is an arbitrary action. If α has no bound names then we have that $bn(\alpha) = \varnothing$ and $z \notin n(\alpha)$ iff $z\#\alpha$. The set of free variables of p is formed only from $\{\alpha\}$ (because P and $P'$are bound by the quantifier $\forall$ and z is bound by *new*), and hence the set of free variables of p is contained in the set $\{z, \alpha\}$; if we assume the free variables of p to be only the variables of p unbound globally, then these free variables of p would be z,α, and again these are contained in the set $\{z, \alpha\}$. We have, because of Proposition 2.23 and Remark 2.24, that "$\forall z.(z\#\alpha$ implies $p)$" if and only if "$И z.p$". Since α was chosen arbitrary we have then the rule $L\text{-}RES_{ln}$ in Table 3 is identical, in this case, with rule number 4 in the nominal semantics of the π-calculus (we assumed that in the set of rules 1-14 in the nominal semantics of the π-calculus, a transition of the form $P \xrightarrow[n]{\alpha} P'$ means a transition without bound names). Now, let us assume that the action has a bound name. For example, let x be the bound name of the action denoted with β. We study two cases: x can be bound by *new* or by *input*. Let q be the formula "$\forall P, P'.(P \xrightarrow[n]{x\,bn\,\beta} P'$ implies $new[z]P \xrightarrow[n]{x\,bn\,\beta} new[z]P')$" (where bn means the binding by *new*) and r be the formula "$\forall P, P'.(P \xrightarrow[n]{y(x)} P'$ implies $new[z]P \xrightarrow[n]{y(x)} new[z]P')$" (where x in bound by *input*). The free variables of q are contained in the set $\{x, z, \beta\}$. More precisely, the set of free variables of q is formed only by $fn(\beta)$ because P, P' are bound by the quantifier $\forall$ and x, z are bound by *new*; if we assume the free variables of q to be only the variables of q unbound globally, then these free variables of q would be x,z, β, and again these are contained in the set $\{x, z, \beta\}$. The free variables of r are contained in the set $\{x, y, z\}$. More exactly, the set of free variables of r is formed

only by y because P, P' are bound by the quantifier $\forall$ and x is bound by *input* and z is bound by *new*; if we assume the free variables of r to be only the variables of r unbound globally, then these free variables of r would be x,y, z, and again these are contained in the set $\{x, y, z\}$. Now, because of Proposition 2.23 and Remark 2.24, we obtain the equivalence results: "$\forall z.(z\#\{x, \beta\}$ implies $q)$" if and only if "$И z.q$", and respectively "$\forall z.(z\#\{x, y\}$ implies $r)$" if and only if "$И z.r$". Since x, y and β were chosen arbitrarily and because for the action α we have $n(\alpha) = \{x\} \cup fn(\alpha)$ i.e. $n(\alpha) = \{x\} \cup supp(\alpha)$, we can say that the rule *L-RES*$_{ln}$ in Table 3 is exactly rule number 5 in the nominal semantics of the π-calculus (point (a) or (b)) as x is bound by *new* respectively by *input*). □

Lemma 4.13. *Rule L-CLOSE*$_{L_{ln}}$ *in Table 3 is identical with rule number 11 in the nominal semantics of the π-calculus.*

Proof. Let p be the formula: "$\forall Q, P', Q', x, y.[(P \xrightarrow[n]{x(y)} P'$ *and* $Q \xrightarrow[n]{\overline{x}(z)} Q')$ implies $P \mid Q \xrightarrow[n]{\tau} new[z](P'\{z|y\} \mid Q')]$". The free variables of p are contained in the set $\{z, P\}$. Indeed, the set of free variables of p is formed only by P (precisely only by the set of free names of P, which is included in P) because $P', Q, Q'x, y$ are bound by the quantifier $\forall$ and z is bound by *new*; if we assume the free variables of p to be only the variables of p unbound globally, then these free variables of p would be z,P, and again these are contained in the set $\{z, P\}$. Now, because of Proposition 2.23 and Remark 2.24, we obtain the equivalence result: "$\forall z.(z\#P$ implies $p)$" if and only if "$И z.p$". Since P was chosen arbitrarily we can say that the rule *L-CLOSE*$_{L_{ln}}$ in Table 3 is exactly rule number 11 in the nominal semantics of the π-calculus. □

We prove the correspondence for replication in a similar way:

Lemma 4.14. *Rule L-REP*$_{close_{ln}}$ *in Table 3 is identical with rule number 14 in the nominal semantics of the π-calculus.*

We give an important result which says that the rules in the nominal semantics of the π-calculus and the rules in Table 3 coincide. Therefore, *when we invoke a rule in the nominal semantics of the π-calculus we tacitly assume its related rule in Table 3 in view of Lemmas 4.9, 4.10, 4.11, 4.12, 4.13, 4.14.*

Corollary 4.15. *The labeled transition rules of Table 3 also represent the* ***nominal semantics*** *of the π-calculus.*

Proof. Since each transition rule in the nominal semantics of the π-calculus described in Figure 2 can be put in a late-nominal form (Lemmas 4.9, 4.10, 4.11, 4.12, 4.13, 4.14) we have that each transition rule of Figure 2 is identical with a rule in Table 3. □

Let the *restricted nominal semantics of the π-calculus* be the semantics where actions are the same as in the nominal semantics of the π-calculus and transition rules are the rules 1-10 and 12-13 in Definition 4.5 (or the rules in Table 3 without *L-CLOSE*$_{L_{ln}}$ and *L-REP*$_{close_{ln}}$). Let the *restricted late semantics of the π-calculus*

be the semantics where the actions are the same as in the late semantics of the π-calculus and the transition rules are the rules presented in Table 2 except $L\text{-}CLOSE_L$ and $L\text{-}REP_{close}$.

We denote by $P \xrightarrow[rn]{\alpha} Q$ a transition of P in Q obtained by applying some transition rules in the restricted nominal semantics of the π-calculus. In fact $P \xrightarrow[rn]{\alpha} Q$ is a transition in the nominal semantics obtained by applying several rules in Figure 2 except rules number 11 and 14. Thus, $\xrightarrow[rn]{\alpha}$ is in fact a $\xrightarrow[n]{\alpha}$- transition. $\xrightarrow[rn]{\alpha}$ does not mean a different transition and does not provide a truly different semantics. Transition of the form $\xrightarrow[rn]{\alpha}$ are obtained by applying $\xrightarrow[n]{\alpha}$-rules. We make a *convention of notation* and we write $\xrightarrow[rn]{\alpha}$ for the **nominal transitions** obtained by applying the rules in restricted nominal semantics only to emphasize that we have some *nominal transitions obtained without using rules 11 and 14*. A similar convention is made for the restricted late semantics. Also, we denote by $P \xrightarrow[rl]{\alpha} Q$ a transition of P in Q obtained by applying some transition rules in the restricted late semantics of the π-calculus. We are able to give a result which expresses that there is a strong connection (but not yet an equivalence) between the late and the nominal semantics of the π-calculus. The following result allows us to say that there is an equivalence between the *restricted nominal semantics of the π-calculus and the restricted late semantics of the π-calculus.*

We now present some theoretical results to show how we can prove the equivalence between the two semantics of the π-calculus. For proving that two semantics of the π-calculus, π_1 (where a transition is denoted by $\rightarrow_1$, an action is denoted by α_1 and a prefix is denoted by p_1) and π_2 (where a transition is denoted by $\rightarrow_2$, an action is denoted by α_2 and a prefix is denoted by p_2) are *"equivalent"* or *"have the same expressive power",* we should be able to define a morphism $\varphi : \pi_1 \rightarrow \pi_2$ with a special property between the syntactic constructions, and to find a way to prove that everything that can be expressed with the transition rules in π_1 can also be expressed with the transition rules in π_2 as an image under the morphism φ and vice versa.

A way to do this is by defining the morphism φ inductively by the rules:

1. $\varphi(0) = 0$.
2. $\varphi(p_1^i.P_1) = p_2^i.\varphi(P_1)$ for each prefix p_1^i and each process P_1, where $\{p_1^i\}$ are prefixes in π_1 and each of the prefixes p_1^i has a corespondent p_2^i in π_2.
3. $\varphi(P_1|Q_1) = \varphi(P_1)|\varphi(Q_1)$ for each P_1, Q_1.
4. $\varphi(new\, xP_1) = new\, x\varphi(P_1)$ for each x, P_1.

With φ defined in this way we should be able to prove that φ is surjective and its kernel is precisely the α-equivalence on π_1. To prove that π_1 and π_2 are equivalent semantics of π we must show (the classical procedure is by induction on the depth of the deduction tree) the following implications:

- For each P_1, Q_1, α_1 in π_1 we have that $P_1 \xrightarrow{\alpha_1}_1 Q_1$ implies $\varphi(P_1) \xrightarrow{\alpha_2}_2 \varphi(Q_1)$, where α_2 is the similar corespondent in π_2 of α_1 (for example if π_1 is the (restricted) late semantics of the π-calculus and π_2 is the (restricted) nominal

semantics of the π-calculus then: if $\alpha_1 = x(y)$ then $\alpha_2 = x(y)$, if $\alpha_1 = \overline{x}(y)$ then $\alpha_2 = y\, bn\, \overline{x}\langle y\rangle$, if $\alpha_1 = \overline{x}\langle y\rangle$ then $\alpha_2 = \overline{x}\langle y\rangle$, and if $\alpha_1 = \tau$ then $\alpha_2 = \tau$).

- For each P_2, Q_2, α_2 in π_2, choosing fresh names for the bound atoms in P_2 and Q_2 such that $P_2 = \varphi(P_1)$ and $Q_2 = \varphi(Q_1)$, we have that $P_2 \xrightarrow{\alpha_2}_2 Q_2$ implies $P_1 \xrightarrow{\alpha_1}_1 Q_1$, where α_2 is the similar corespondent in π_2 of α_1 (for example if π_2 is the (restricted) nominal semantics of the π-calculus and π_1 is the late semantics of the π-calculus then: if $\alpha_2 = x(y)$ then $\alpha_1 = x(y)$, if $\alpha_2 = y\, bn\, \overline{x}\langle y\rangle$ then $\alpha_1 = \overline{x}(y)$, if $\alpha_2 = \overline{x}\langle y\rangle$ then $\alpha_1 = \overline{x}\langle y\rangle$, and if $\alpha_2 = \tau$ then $\alpha_1 = \tau$).

In our case, for the restricted semantics of the π-calculus (late and nominal), things are very clear. Both in the restricted late semantics of the π-calculus and in the restricted nominal semantics of the π-calculus, *the processes are defined by the same syntax and the basic actions are the same.* So, it is not necessary to define explicitly a morphism φ between the restricted late semantics of the π-calculus and the restricted nominal semantics of the π-calculus as in the general theory. We can assume φ *to be a morphism which leaves the processes and the actions in both semantics unchanged (we recall that the terms obtained after an* α*-conversion are identified in the nominal logic) and whose kernel is the* α*-equivalence on the restricted late semantics of the* π*-calculus.*

Such a φ is inductively induced by the nominal abstraction. The main property of φ is that for any process P in π_1 we have $z \notin fn(P)$ iff $z\#\varphi(P)$ (easy to prove by induction on the syntax of φ). These properties of φ allow us to make the convention of eliminating φ in the proof of the following theorem (φ is seen as *inclusion* for easy writing).

Theorem 4.16. *For any two processes* P *and* Q *and action* α*, we have the equivalence:*

$$P \xrightarrow[rl]{\alpha} Q \text{ if and only if } P \xrightarrow[rn]{\alpha} Q.$$

Proof. We recall that $\xrightarrow[rn]{\alpha}$ are obtained by applying $\xrightarrow[n]{\alpha}$-rules. rn does not have a special meaning as l or n have. rn- transitions are in fact n-transitions. We made a *convention of notation* and we write $\xrightarrow[rn]{\alpha}$ for the nominal transitions obtained by applying the rules in restricted nominal semantics only to emphasize that we have some *nominal transitions obtained without using the rules 11 and 14.* Also, by analogy, the $\xrightarrow[rl]{\alpha}$ are obtained by applying $\xrightarrow[l]{\alpha}$-rules. For the restricted nominal semantics of the π-calculus we consider the set of transition rules presented in Table 3 without $L\text{-}CLOSE_{L_{ln}}$ and $L\text{-}REP_{close_{ln}}$instead of the transition rules presented in Figure 2 without rules 11 and 14. By corollary 4.15, these two sets of transition rules coincide. Each transition of form $P \xrightarrow[rn]{\alpha} Q$ is obtained by repeatedly applying the basic transition rules presented in Table 3 without $L\text{-}CLOSE_{L_{ln}}$ and $L\text{-}REP_{close_{ln}}$. We have to prove a double implication. The first part is proved by induction after the depth of the deduction tree of $\xrightarrow[rl]{}$, the second by induction after the depth of the deduction tree of $\xrightarrow[rn]{}$ To save space we do not rewrite out the induction hypotheses

in complete formality. We analyze only few cases. The rest of them are similar or trivial. First we prove that $P \xrightarrow[rl]{\alpha} Q$ implies $P \xrightarrow[rn]{\alpha} Q$.

Case *L-PAR*$_L$. Let us assume that $\alpha = x(y)$ (the case when $\alpha = \overline{x}(y)$ is identical and the case when $\alpha = \tau$ is trivial). Suppose $P \xrightarrow[rl]{x(y)} P'$ and $y \notin fn(Q)$ for some Q. Then $P \mid Q \xrightarrow[rl]{x(y)} P' \mid Q$. By the inductive hypothesis $P \xrightarrow[rn]{x(y)} P'$. Since, in the FM approach, $y \notin fn(Q)$ is the same with $y\#Q$ (precisely with $y\#\varphi(Q)$; however we made a convention to eliminate φ for easy writing) we can apply the rule PAR_{ln} and we obtain $P \mid Q \xrightarrow[rn]{x(y)} P' \mid Q$.

Case *L-OPEN*. Let us assume that $P \xrightarrow[rl]{\overline{x}\langle z\rangle} P'$ and $z \neq x$. Then $new\, zP \xrightarrow[rl]{\overline{x}(z)} P'$. By the inductive hypothesis $P \xrightarrow[rn]{\overline{x}\langle z\rangle} P'$. Since in the FM approach, $z \neq x$ means $z\#x$ we can apply *L-OPEN*$_{ln}$ and we obtain $new[z]P \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} P'$.

Case *L-RES*. Let us assume that $\alpha = \overline{x}(y)$ (the case when $\alpha = x(y)$ is identical and the case when $\alpha = \tau$ is trivial). Suppose $P \xrightarrow[rl]{\overline{x}(y)} P'$ and $z \notin n(\overline{x}(y))$. Then $new\, zP \xrightarrow[rl]{\overline{x}(y)} new\, zP'$. By the inductive hypothesis $P \xrightarrow[rn]{y\, bn\, \overline{x}\langle y\rangle} P'$. Since $z \notin n(\overline{x}(y))$ means $z \neq x, y$ which is $z\#\alpha$ and $z \notin bn(\alpha)$ ($bn(\alpha)$ is only a notation in the sense of Remark 4.8), we can apply the rule *L-RES*$_{ln}$ and we obtain $new[z]P \xrightarrow[rn]{y\, bn\, \overline{x}\langle y\rangle} new[z]P'$.

We prove the second implication which is: $P \xrightarrow[rn]{\alpha} Q$ implies $P \xrightarrow[rl]{\alpha} Q$.

Case *L-PAR*$_{L_{ln}}$. Let us assume that $\alpha = x(y)$ (the case when $\alpha = y\, bn\, \overline{x}\langle y\rangle$ is identical and the case when $\alpha = \tau$ is trivial). Suppose $P \xrightarrow[rn]{x(y)} P'$ and $y\#Q$ for some Q. Then $P \mid Q \xrightarrow[rn]{x(y)} P' \mid Q$ by *L-PAR*$_{L_{ln}}$. By the inductive hypothesis $P \xrightarrow[rl]{x(y)} P'$. Since $y\#Q$ means $y \notin supp(Q) = fn(Q)$, we can apply the rule *L-PAR*$_L$ and we obtain $P \mid Q \xrightarrow[rl]{x(y)} P' \mid Q$.

Case *L-OPEN*$_{ln}$. Let us assume that $P \xrightarrow[rn]{\overline{x}\langle z\rangle} P'$and $z\#x$. Then we have that $new[z]P \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} P'$. By the inductive hypothesis $P \xrightarrow[rl]{\overline{x}\langle z\rangle} P'$. Since in the FM approach, $z\#x$ means $z \notin supp(x) = \{x\}$ we can apply *L-OPEN* and we obtain $new\, zP \xrightarrow[rl]{\overline{x}(z)} P'$.

Case *L-RES*$_{ln}$. Let us assume that $\alpha = y\, bn\, \overline{x}\langle y\rangle$ (the case when $\alpha = x(y)$ is identical and the case when $\alpha = \tau$ is trivial). Suppose $P \xrightarrow[rn]{y\, bn\, \overline{x}\langle y\rangle} P'$ and $z\#(y\, bn\, \overline{x}\langle y\rangle)$, $z \notin bn(y\, bn\, \overline{x}\langle y\rangle)$. Then $new[z]P \xrightarrow[rn]{y\, bn\, \overline{x}\langle y\rangle} new[z]P'$ by *L-RES*$_{ln}$. By the inductive hypothesis $P \xrightarrow[rl]{\overline{x}(y)} P'$. Since $z\#(y\, bn\, \overline{x}\langle y\rangle)$ and $z \notin bn(y\, bn\, \overline{x}\langle y\rangle)$ (see Remark 4.8!)

means $z \notin supp(\overline{x}(y)) = \{x\}$ and $z \notin \{y\}$ we have that $z \notin n(\overline{x}(y))$. So, we can apply the rule *L-RES* and we obtain $new\, zP \xrightarrow[rl]{\overline{x}(y)} new\, zP'$.

The proof is now complete. □

Remark 4.17. *If we apply, step by step, the method of proving the equivalence of several semantics presented before, in the proof of the Theorem 4.14 we obtain in fact (by induction) that* $P \xrightarrow[rn]{\alpha} Q$ *implies* $P^{'} \xrightarrow[rl]{\alpha} Q^{'}$, *where* $P = \varphi(P^{'})$ $Q = \varphi(Q^{'})$ *for fresh choices of bound names in* P *and respectively in* Q. *Since* φ leaves the processes and the actions in both semantics unchanged, *we have that* $P = \varphi(P)$ $Q = \varphi(Q)$. *Now, because the kernel of* φ *is precisely the* α*-equivalence on the restricted late semantics of the* π*-calculus, it follows that* $P \equiv_\alpha P^{'}$ *and* $Q \equiv_\alpha Q^{'}$. *Now, from [13], it is clear that* $P \xrightarrow[rl]{\alpha} Q$.

In view of Theorem 4.16 we are able to say that *all the rules in the late semantics of the* π*-calculus, except* $L\text{-}CLOSE_L$ *and* $L\text{-}REP_{close}$ *are equivalent (or can be identified) with those in the nominal semantics of the* π*-calculus, except rules number 11 and 14,* which are identical with those in Table 3, except $L\text{-}CLOSE_{L_{ln}}$ and $L\text{-}REP_{close_{ln}}$.

The main differences between the nominal semantics of the π-calculus and the late semantics of the π-calculus are, now, the rules number 11 and 14, which seems to do not be equivalent with any rules in the late semantics of the π-calculus.

So why we chose to present the rules number 11 and 14 in the form we presented them in Figure 2?

First let us consider the transition rules:

11': $$\frac{P \xrightarrow[n]{x(z)} P', Q \xrightarrow[n]{\overline{x}(z)} Q'}{P \mid Q \xrightarrow[n]{\tau} new[z](P' \mid Q')}$$

14': $$\frac{P \xrightarrow[n]{\overline{x}(z)} P', P \xrightarrow[n]{x(z)} P''}{!P \xrightarrow[n]{\tau} (new[z](P' \mid P'')) \mid !P}$$

Let the *trivial nominal semantics of the* π*-calculus* be the semantics of the π-calculus defined by the transition rules 1-10, 12-13 in Figure 2 and 11', 14' considered before. A transition of the form $P \xrightarrow{u} Q$ (where u is the action) in the trivial nominal semantics of the π-calculus is denoted by $P \xrightarrow[tn]{u} Q$.

It is obvious to remark that the *late semantics* and the *trivial nominal semantics* of the π-calculus have the same expressive power because, exactly in the same way we proved Theorem 4.16, we can prove the following result:

Theorem 4.18. *For any two processes* P *and* Q *and action* α, *we have the equivalence:*

$$P \xrightarrow[tn]{\alpha} Q \text{ if and only if } P \xrightarrow[l]{\alpha} Q.$$

So we can define a semantics of the π-calculus, with transition rules expressed in a nominal form (using the nominal quantifier), which is **equivalent** with the late

semantics of the π-calculus. In [1] the authors have provided a nominal semantics of the πI-calculus which is equivalent with the usual Sangiorgi semantics of the πI-calculus presented in [14]. In this paper we could have tried a similar approach as in [1]. Anyway the techniques developed in this paper are similar which those developed in [1]. In fact we have already presented a similar result as in [1] which says that the trivial nominal semantics and the late semantics of the π-calculus are equivalent (Theorem 4.18). However we intend to have a deeper approach of the topic.

We do not want just to rewrite the transition rules of the late semantics of the π-calculus in a compact nominal form in the same way we did in [1] with the transition rules of the Sangiorgi semantics of the πI calculus. We want to use the nominal quantifier to express the transition rules of the π-calculus in the "weakest" form we need for describing the mobility mechanism. Precisely:

The rule number 11' is assumed to be valid for all z (even for those z which are in $fn(P)$). According to Proposition 4.21 and Remark 4.23, for describing the mobility it is enough to assume that a transition rule, similar with 11', is valid only for those z which are not in $fn(P)$. So for describing the mobility we do not need an entire rule 11' valid for all z, but a rule 11 which must be valid only for some z satisfying a freshness condition. Actually the rule number 11 in the nominal semantics of the π-calculus is motivated by the manner in which the links are moved in a virtual space of linked processes.

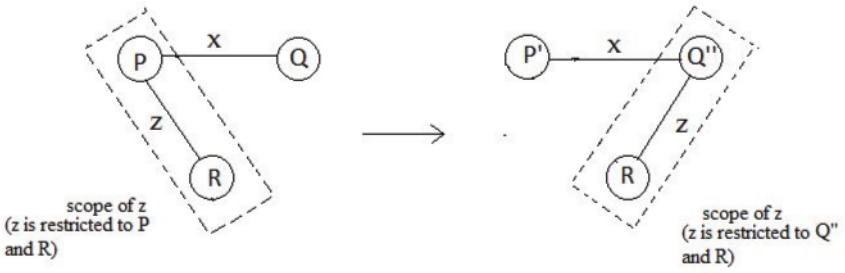
If we had wanted to define a semantics of the π calculus, expressed by using nominal techniques, which is equivalent with the late semantics of the π-calculus, we should have defined the nominal semantics of the π-calculus to be the trivial nominal semantics of the π-calculus. However we want something more than a simple rewriting in nominal terms of the transition rules in the late semantics of the π-calculus (which, anyway, would have been interesting). We make a refinement of the transition rule $L\text{-}CLOSE_L$ such that we assume this rule to be valid only for those z which are fresh for P (and not for all z as in the late semantics of the π-calculus) because for describing the mobility this is weakest condition requested in the rule's hypothesis (Remark 4.23). The nominal quantifier helps us to "encode" the condition "z fresh for P" in the hypothesis of the rule number 11 such that this rule is presented in a compact form. Even now we are tempted to say that the rules 11' (which is similar to $L\text{-}CLOSE_L$ and is valid for all z) and 11 (which is induced by the manner in which the links are moved and, so, it is assumed to be valid only for some z) are completely different, the results developed in Section 5 allow us to establish a strong connection between them.

Remark 4.19. *Theorem 3.10 remains valid in the nominal approach if we replace in its statement e with n. The proof is actually the same as in [13]. By induction on the inference of* $P \xrightarrow[n]{\tau} P'$ *we show (in the same way as in [13]) that for any two processes* P, P' *we have that* $P \xrightarrow[n]{\tau} P'$ *implies* $P \longrightarrow P'$.

In the following example we describe formally the mobility mechanism as in [9].

Example 4.20. We refer to the example provided in [9] in order to explain mobility and how links can move between two processes. If $P = \overline{x}\langle z\rangle.P'$ where z is

not free in $P^{'}$, z is not free in Q (Q cannot be linked by z) and $Q = x(y).Q'$, then $new\, z(P|R)|Q \longrightarrow P^{'}|new\, z(R|Q^{''})$ where $Q^{''} = Q^{'}\{z|y\}$ (we keep the same notations as in [9]). First we show that we can prove that this mobility is true by using the labeled transition rules of the nominal semantics, which correspond to some of those in the late semantics of the π-calculus. We know that $P \xrightarrow[n]{\overline{x}\langle z\rangle} P^{'}$ and, by rule number 2, that $P|R \xrightarrow[n]{\overline{x}\langle z\rangle} P^{'}|R$. We also have $Q \xrightarrow[n]{x(y)} Q^{'}$ and, because of rule number 10 we obtain that $(P|R)|Q \xrightarrow[n]{\tau} (P^{'}|R)|Q^{''}$. Now, because the internal action τ has neither free names nor bound names, we can apply the rule number 4 ($z \notin fn(\tau) = \varnothing$, *i.e.* $z\#\tau$) and we obtain $new[z]((P|R)|Q) \xrightarrow[n]{\tau} new[z]((P^{'}|R)|Q^{''})$. From Remark 4.19 we get $new\, z((P|R)|Q) \longrightarrow new\, z((P^{'}|R)|Q^{''})$. However in the simple operational semantics of the π-calculus a structural congruence is defined (Definition 3.6) We know that, for two processes S_1 and S_2 in the simple operational semantics of the π-calculus, if $y \notin fn(S_1)$ then $new\, y(S_1|S_2) \equiv S_1|new\, yS_2$. In our case $z \notin fn(P^{'})$ (P loses the link to R), $z \notin fn(Q)$, $(P|R)|Q \equiv Q|(P|R)$, $(P^{'}|R)|Q^{''} \equiv P^{'}|(R|Q^{''})$ and so, by the rule **str** in Definition 3.7, we obtain the transition $new\ z(P|R)|Q \longrightarrow P^{'}|new\ z(R|Q^{'}\{z|y\})$. An intuitive graphic representation of this link movement is described in the next figure.



Proposition 4.21. *The freshness condition assumed in the hypothesis of the rule number 11 of the nominal semantics of the π-calculus is motivated by the manner in which the links are moved in a virtual space of linked processes.*

Proof. In the view of Lemma 4.13, rule number 11 in the nominal semantics of the π-calculus can be identified with $L\text{-}CLOSE_{L_{ln}}$ in Table 3. Let us analyze again the standard example which defines mobility, respectively Example 4.20. If $P = \overline{x}\langle z\rangle.P^{'}$ where z is not free in $P^{'}$, z is not free in Q (Q cannot be linked by z) and $Q = x(y).Q'$, then $new\, z(P|R)|Q \longrightarrow P^{'}|new\, z(R|Q^{''})$ where $Q^{''} = Q^{'}\{z|y\}$ (we keep the same notations as in [9]). We prove that this mobility is true using the labeled transition rules of the nominal semantics. We know that $P \xrightarrow[n]{\overline{x}\langle z\rangle} P^{'}$ and, by rule number 2, that $P|R \xrightarrow[n]{\overline{x}\langle z\rangle} P^{'}|R$. We also assume that $x \neq z$ (we assume that P and R, respectively P and Q have distinct links because z is restricted to P and

R) and so, from rule number 8 we get $new[z](P|R) \xrightarrow[n]{z\,bn\,\overline{x}\langle z\rangle} P^{'}|R$. From this transition, and because $Q \xrightarrow[n]{x(y)} Q^{'}$, we get by rule number 11 that $Q|(new[z](P|R)) \xrightarrow[n]{\tau} new[z](Q^{'}\{z|y\}|(P^{'}|R))$. We could apply rule number 11 **because $z\#Q$**. From Remark 4.19 we have $Q|(new\ z(P|R)) \longrightarrow new\ z(Q^{'}\{z|y\}|(P^{'}|R))$ A structural congruence is defined in the simple operational semantics of the π-calculus. We know that, for two processes S_1 and S_2 in the simple operational semantics of the π-calculus, if $y \notin fn(S_1)$ then $new\ y(S_1|S_2) \equiv S_1|new\ yS_2$. In our case $z \notin fn(P^{'})$ (P loses the link to R), $z \notin fn(Q)$, $(P|R)|Q \equiv Q|(P|R)$, $(P^{'}|R)|Q^{''} \equiv P^{'}|(R|Q^{''})$ and so, by rule **str** in Definition 3.7, we have $new\,z(P|R)|Q \longrightarrow P^{'}|new\,z(R|Q^{'}\{z|y\})$. We have also presented the reason for which we chose to expose rule number 11 only for those z which does not belong to $fn(P)$ (we keep the notations in Definition 4.5). To move a link, as in the standard example for mobility presented before, we need that z is initially fresh for the process which is not linked by z (z is fresh for Q because initially Q is not linked with some other process by z; after a transition, Q will be connected with other process by a link labeled with z, and in this way, the link labeled with z is moved). As a result, in order to express link movement, we need to present rule number 11 only for those z which are fresh for P (notations are those in Definition 4.5). An entire rule, corresponding to $L\text{-}CLOSE_L$, of the form $\forall P, z, Q, R, S, x, y.\dfrac{P \xrightarrow[n]{x(y)} R\ and\ Q \xrightarrow[n]{z\,bn\,\overline{x}\langle z\rangle} S}{P|Q \xrightarrow[n]{z\,bn\,\tau} new[z]((R\{z|y\})|S)}$ *(without freshness conditions on z)* would be superfluous. Moreover, the part of this rule called $\forall P, Q, R, S, x, y.\forall z \in fn(P).\dfrac{P \xrightarrow[n]{x(y)} R\ and\ Q \xrightarrow[n]{z\,bn\,\overline{x}\langle z\rangle} S}{P|Q \xrightarrow[n]{z\,bn\,\tau} new[z]((R\{z|y\})|S)}$, cannot be applied to explain the link movement. □

It is enough to assume that rule number 11 is valid only in the restrictive hypothesis $z\#P$ (we keep the notations from Definition 4.5).

Now we give some intuitive reasons why it is also necessary to assume that $z\#P$ in the hypothesis of rule number 11, in order to describe the mobility.

All the rules of the nominal semantics, except two of them, are similar to those of the late semantics of the π-calculus, as we proved before. The mentioned exceptions are given by rules number 11 and 14. The difference from the late semantics of the π-calculus is that, in our approach, we assume that the bound variable z (using the notation of rule number 11) is fresh for process P. This approach is justified by previous example which show how the mobility can be expressed. If we could not assume (keeping the notations in Example 4.20) that z is fresh, then we should accept that P and Q, respectively P and R (with the notations from the previous example) could be linked by a channel labeled each time with z. However this is something we want to avoid. We try to analyze the mobility mechanism i.e. the case when a link between two processes is moved to a link between another two processes after a transition. If all the processes (in our case P, Q and R) use the same z to

communicate then there is no link movement between them. Thus we could not move the link between P and R into a link between Q and R. Rule number 11 is enough to present the mobility part of the π-calculus. The case $z \in fn(P)$ (with notations of rule number 11) would not be interesting, and a rule 15:

$\forall P, Q, R, S, x, y. \forall z \in fn(P). \dfrac{P \xrightarrow[n]{x(y)} R \, and \, Q \xrightarrow[n]{z \, bn \, \overline{x}\langle z\rangle} S}{P|Q \xrightarrow[n]{z \, bn \, \tau} new[z]((R\{z|y\})|S)}$ would not be useful to express mobility. In our vision such a rule (15) would be useless. For example, if $Q = new[z]Q'$ and $Q' \xrightarrow[n]{\overline{x}\langle z\rangle} S$ then, if we want to move a link (labeled with z) between Q' and a certain process W into a link between a descent of P and W, we return to Example 4.20. By the proof of Proposition 4.21 it is enough to give the rule number 11 in the form we presented it in Figure 2. Let us try to analyze what happens if we want to move a link (labeled with z) between P and a certain process V. Moreover, if we would accept rule number 15, because $z \in fn(P)$ we would have (intuitively) the possibility that there is a process V such that P and V are linked by a channel labeled with z. We want to see how the link between P and V is removed. The case when $z \in fn(Q)$ and each of P, V and Q use z to communicate does not involve a link movement. However Q outputs a bound name (see also Proposition 5.1 point 5) which was bound by *new*. It follows that z is restricted to some components of Q which use a channel labeled by z to communicate. The channel between P and Q is labeled by x. In our hypothesis (when we want to remove the link labeled by z between P and V) we have that z is used by P and V. However z cannot be restricted to P and V since z was already restricted to some components of Q. Again, the desired link movement does not appear.

The following result is now trivial (see Definition 4.5 and Proposition 4.21):

Corollary 4.22. *Mobility can be described in the nominal semantics of the π-calculus by using rule number 11, which is assumed to be valid only in a more restrictive hypothesis than its related rule L-$CLOSE_L$ in the late semantics of the π-calculus.*

Remark 4.23. *We remark from the previous corollary (in fact from the proof of Proposition 4.21) that, for describing mobility, it is enough to assume that rule number 11 is valid only for those z which are not in $fn(P)$ (we keep the notations from Definition 4.5). Its related rule L-$CLOSE_L$ is assumed to be valid for all z and not only for those which do not belong to $fn(P)$.*

Informally we make a refinement of the rule L-$CLOSE_L$, and we present it in the "weakest" form we need for describing the "link movement" because, in order to describe the mobility, we do not need to assume a rule as L-$CLOSE_L$ which is valid for all z but only a similar rule which is valid for some z satisfying a freshness condition. In section 6 we prove that even if we introduce the nominal quantifier to make a refinement of L-$CLOSE_L$ in the nominal semantics of the π-calculus, the new semantics provides actually the same transitions as the late semantics of the π-calculus.

5. Binding Operators in the Nominal π-calculus

Some other benefits for the introduction of the nominal semantics of the π-calculus with the transition rules expressed in a compact form (by using quantifiers and not supplementary conditions in the hypotheses of the transition rules) are presented in this section, where we are able to study easier the behavior of *new* and *input* when they act together in the same rule. Some results given (for example in [13]) for the late semantics of the π-calculus have corespondents in the nominal semantics of the π-calculus. In this section we present several properties of nominal transitions which help us to compare the nominal semantics of the π-calculus and the late semantics of late semantics of the π-calculus (Section 6).

Proposition 5.1. *We have the following freshness properties of simple transition rules:*

1. *If* $P \xrightarrow[n]{\overline{x}\langle z\rangle} Q$ *then* $x, z \in supp(P)$ *and* $supp(Q) \subseteq supp(P)$.
2. *If* $P \xrightarrow[n]{x(z)} Q$ *then* $x \in supp(P)$ *and* $supp(Q) \subseteq supp(P) \cup \{z\}$.
3. *If* $P \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} Q$ *then* $x \in supp(P)$ *and* $supp(Q) \subseteq supp(P) \cup \{z\}$.
4. *If* $P \xrightarrow[n]{\tau} Q$ *then* $supp(Q) \subseteq supp(P)$.
5. *If* $P \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} Q$ *and the transition derivation does not contain an application of* $L\text{-}SUM_{L_{ln}}$ *then* $z\#P$.

Proof. The proof can be done by induction on transition rules. Each transition of form $P \xrightarrow[n]{\beta} Q$ can be obtained from the set 1-14 of rules in the Definition 4.5. So, we have to analyze each case of simple transition separately. All inferences are routine, so we give only two examples. The first assertion is needed to prove the third and the first three are needed to prove the last. Assertion number 5 is again an induction on the inference of $P \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} Q$.

In the proof of the third part, let us assume that $P = new[z]R$ and $P \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} Q$ is obtained by applying rule number 8, from $R \xrightarrow[n]{\overline{x}\langle z\rangle} Q$. By the first part $x, z \in supp(R)$ and $supp(Q) \subseteq supp(R)$. Since in the hypothesis of rule number 8 we have $z\#x$ it follows that $x \in supp(P)$ and $supp(Q) \subseteq supp(P) \cup \{z\}$.

In the proof of part 4 let us assume that $P \xrightarrow[n]{\tau} Q$ is obtained by the rule number 11 from $R \xrightarrow[n]{x(y)} R'$ and $S \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} S'$ where $z\#R$. It follows that $R|S \xrightarrow[n]{\tau} new[z]((R'\{z|y\})|S')$. By the third part we have $supp(S') \subseteq supp(S) \cup \{z\}$ and by the second part we obtain $supp(R') \subseteq supp(R) \cup \{y\}$. Let $Q' = (R'\{z|y\})|S'$ and $Q = new[z]((R'\{z|y\})|S')$. Then $P = R|S \xrightarrow[n]{\tau} Q$ and $supp(Q) = supp(Q') - \{z\} \subseteq supp(R|S) = supp(P)$. The proof is complete. □

Proposition 5.2. *Let* M *be finitely generated by substitutions. If* $x\#M$, *then there is only one* y *such that* $y = Mx$, *and* $y = x$.

Proof. Since M is finitely generated by substitutions, there are cofinitely many atoms y such that $\forall z.Mz = y \Rightarrow z = y$ (*). For example, considering a substitution $\{b|a\}$, all the atoms except b have the property of y described before. By the special property of our interchange function, we have cofinitely many y such that $y\#M$, and so we can choose one of these y such that $My = y$ and y has property (*). We have $(x\,y)(Mx) = ((x\,y)\cdot M)((x\,y)\cdot x) \overset{x,y\#M}{=} My = y$, and so $Mx = x$ by the definition of transpositions. Now, if there is an atom z such that $x = Mz$, then $y = (x\,y)\cdot x = M((x\,y)\cdot z)$ and by (*), $(x\,y)\cdot z = y$. It follows that $z = x$. □

Remark 5.3. *The converse of this proposition is also true; this result is proved in [4]. Also, the result presented in Remark 2.22 point 3 is a corollary of Proposition 5.2 as it is proved in [4].*

Theorem 5.4. *Let M be the monoid generated by substitutions of form $\{b|a\}$. Then*

$$\forall P.\text{И}y.\forall \alpha, Q.P \xrightarrow[n]{y\,bn\,\alpha} Q \text{ implies } PM \xrightarrow[n]{y\,bn\,\alpha M} QM.$$

whenever $y\#M$.

Proof. We proceed by induction on the depth of the deduction tree.

For example, let us take the case of rule number 8. Suppose we have x, y, P, Q and $P \xrightarrow[n]{y\,bn\,\overline{x}\langle y\rangle} Q$ is inferred by $L\text{-}OPEN_{ln}$. Suppose that $P = new[y]R$ and $R \xrightarrow[n]{\overline{x}\langle y\rangle} Q$ with $y\#x$. We take advantage of the FM framework to choose y fresh also for M, so that $yM = y$ makes sense. Using a result presented in Remark 2.22 point 3, we know that $y\#xM$ and $y = My$. Now, using another structural induction on the inference of $\xrightarrow[n]{output}$ we have that $RM \xrightarrow[n]{\overline{xM}\langle y\rangle} QM$. For this new set of processes and actions, we may apply rule number 8 and we get $new[y]RM \xrightarrow[n]{y\,bn\,\overline{xM}\langle y\rangle} QM$. Definition 4.4 shows us that $PM \xrightarrow[n]{y\,bn\,\overline{xM}\langle y\rangle} QM$ because $P = new[y]R$ and $PM = new[y]RM$.

Let us now take the case of rule number 3. Suppose that $P \xrightarrow[n]{y\,bn\,\overline{x}\langle y\rangle} Q$ is inferred by $L\text{-}PAR_{L_{ln}}$ from $V \xrightarrow[n]{y\,bn\,\overline{x}\langle y\rangle} W$. Suppose that $P = V|R$, $Q = W|R$ where $y\#R$. We chose y fresh for M. The inductive hypothesis says that $VM \xrightarrow[n]{y\,bn\,\overline{xM}\langle y\rangle} WM$. Also $y\#R$ and $y\#M$ implies $y\#RM$ and we can apply rule 3 point a to get exactly what we want: $VM|RM \xrightarrow[n]{y\,bn\,\overline{xM}\langle y\rangle} WM|RM$. □

Remark 5.5. *In the previous theorem we worked using the hypothesis that $y\#M$ (the bound variable for each case is not in the support of M). In this way, Proposition 5.2 is applied. The necessity of assuming $y\#P$ and $y\#M$ in the hypothesis of the previous theorem is also justified by the following example: let us assume, for example, that $P = new[y]\overline{x}\langle y\rangle.\overline{y}\langle z\rangle.0 + \overline{x}\langle y\rangle$ (the only case when y could belong to $fn(P)$ is when P is obtained by an application of $L\text{-}SUM_{L_{ln}}$ as it is proved in Proposition 5.1) and $\alpha = y\,bn\,\overline{x}\langle y\rangle$. Then $P \xrightarrow[n]{y\,bn\,\overline{x}\langle y\rangle} Q$ where $Q = \overline{y}\langle z\rangle.0$; if $\sigma = \{y|z\}$*

then we cannot have $P\sigma \xrightarrow[n]{y\, bn\, \alpha\sigma} Q\sigma$ *because* y *cannot be the object of a bound output of* PM *(because of capture-free avoidance convention in* π*-calculus, the bound name* y *in* P *has to be renamed when we compose* $P\sigma$*) . In fact this example also show us that in the hypothesis of Theorem 5.4 we have to request a stronger condition which is* $y\#PM$*.*

The possibility of choosing y *fresh for* M *in the proof of Theorem 5.4 is guaranteed by the finite support property in FM.*

Several result of this type can be obtained using nominal techniques. We finish this part with another example of a rule where α-abstraction (symbolized by *new* as we assumed in this part) and substitution act together.

Theorem 5.6. $\forall P, \forall y, \forall x, z, Q.P \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} Q$ *implies* $P \xrightarrow[n]{y\, bn\, \overline{x}\langle y\rangle} Q\{y|z\}$.

Proof. We remark first that the previous theorem remains valid for $y = z$, but the proof is trivial. Now, let us start the proof of our theorem.

We proceed by induction of the inference of $P \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} Q$. Again it is a routine case analysis. We consider only the most important case.

Assume $P = new[z]R$ and $P \xrightarrow[n]{z\, bn\, \overline{x}\langle z\rangle} Q$ is obtained by rule number 8 (in the definition of the nominal semantics of the π- calculus) from $R \xrightarrow[n]{\overline{x}\langle z\rangle} Q$, $z\#x$ (*i.e.* $z \neq x$). Suppose $y\#P$. Then $y\#x$ since $x \in supp(P)$, by Proposition 5.1. Moreover, $P = new[y]R\{y|z\}$. It is trivial (by induction) to remark that $R\{y|z\} \xrightarrow[n]{\overline{x}\langle z\rangle\{y|z\}} Q\{y|z\}$, that is $R\{y|z\} \xrightarrow[n]{\overline{x}\langle y\rangle} Q\{y|z\}$. Now we apply rule number 8 again and we get $P \xrightarrow[n]{y\, bn\, \overline{x}\langle y\rangle} Q\{y|z\}$. Tacitly we applied Proposition 2.23 and Remark 2.24 in the same way we did in proof of Lemmas 4.9, 4.10, 4.11 or 4.12. □

Theorem 5.7. $\forall P, \forall y, \forall x, z, Q.P \xrightarrow[n]{x(z)} Q$ *implies* $P \xrightarrow[n]{x(y)} Q\{y|z\}$.

Proof. Let p be the formula: "$\forall x, z, Q.P \xrightarrow[n]{x(z)} Q$ implies $P \xrightarrow[n]{x(y)} Q\{y|z\}$". The free variable of p are contained in the set $\{y, P\}$. Indeed, the set of free variables of p is formed only by P (precisely only by the set of free names of P, which is included in P) because x, z, Q are bound by the quantifier $\forall$ and z is bound by *input*; if we assume the free variables of p to be only the variables of p unbound globally, then these free variables of p would be y,P, and again these are contained in the set $\{y, P\}$. Now, because of Proposition 2.23 and Remark 2.24, we obtain the equivalence result: "$\forall y.(y\#P$ implies $p)$" if and only if "$\forall y.p$". Since P was chosen arbitrarily, the statement of this theorem is equivalent with "$P \xrightarrow[n]{x(z)} Q$ implies $P \xrightarrow[n]{x(y)} Q\{y|z\}$ whenever $y\#P$". Now suppose $y\#P$. Then $y\#x$ since $x \in supp(P)$ (Proposition 5.1) and the transition labeled by $x(y)$ have sense.

We suppose $y \neq z$ (the case $y = z$ is trivial). We proceed by induction on the inference of $P \xrightarrow[n]{x(z)} Q$. It is a routine case analysis. The most important case is when

$P \xrightarrow[n]{x(z)} Q$ is obtained from the rule number 1 in Figure 2. So let $P = x[z]Q$. According to Corollary 2.29 and Proposition 5.1 we have $supp(P) = (supp(Q) \setminus \{z\}) \cup \{x\}$. Let $y\#P$. This means $y \notin supp(P)$ and hence $y \notin supp(Q)$. Since $y\#Q$ we have $P = x[y]Q\{y|z\}$. By rule number 1 in Figure 2 we have $P = x[y]Q\{y|z\} \xrightarrow[n]{x(y)} Q\{y|z\}$.

We analyze now (for example) the case when $P \xrightarrow[n]{x(z)} Q$ is obtained from the rule number 3 in Figure 2 (or equivalently from the rule $L\text{-}PAR_{L_{ln}}$ in Table 3). Let $P = P_1|R$ and $Q = Q_1|R$ such that $z\#R$ and $P_1 \xrightarrow[n]{x(z)} Q_1$. Clearly $supp(P) = supp(P_1) \cup supp(R)$. If $y\#P$ we also have $y\#P_1$ and $y\#R$. The inductive hypothesis says that $P_1 \xrightarrow[n]{x(y)} Q_1\{y|z\}$. However $z\#R$ and $y\#R$. Hence $y\#R\{y|z\}$. We can apply the rule number 3 in Figure 2 and we obtain $P_1|R\{y|z\} \xrightarrow[n]{x(y)} Q_1\{y|z\}|R\{y|z\}$. From Definition 4.3 we have that $Q_1\{y|z\}|R\{y|z\} = Q\{y|z\}$. Also, because $z\#R$, we have $R\{y|z\} = R$. Finally we obtain $P \xrightarrow[n]{x(y)} Q\{y|z\}$. □

Remark 5.8. *Using another induction on the depth of the deduction tree for the trivial nominal semantics of the π-calculus (whose transitions are expressed by the rules 1-10,11',12-13 and 14'), we can prove that all the results presented in this section are also valid in the trivial nominal semantics of the π-calculus. A similar argument shows us that the results in this section are valid even in the restricted nominal semantics of the π-calculus. Moreover, a transition of form $P \xrightarrow[n]{u} Q$ is equivalent with $P \xrightarrow[rn]{u} Q$ whenever the action u is different from τ (the rules number 11 and 14 which make the difference between the restricted nominal semantics and the nominal semantics of the π-calculus provide only transitions labeled by the internal action τ).*

6. Equivalence between Nominal and Late Semantics of the π-calculus

The results presented in Section 5 allow us to make a comparison between the trivial nominal semantics and the nominal semantics of the π-calculus. We prove that the rule number 11 in the nominal semantics of the π-calculus is equivalent with the rule number 11' in the trivial nominal semantics of the π-calculus. Analogue the rule number 14 in the nominal semantics of the π-calculus is equivalent with the rule number 14' in the trivial nominal semantics of the π-calculus.

Lemma 6.1. *In the restricted nominal semantics of the π-calculus the following sentences are equivalent:*

p: *"If* $P \xrightarrow[rn]{x(z)} R$ *and* $Q \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} S$ *then* $P \mid Q \xrightarrow[rn]{\tau} new[z](R \mid S)$*"*

q: *"If* $P \xrightarrow[rn]{x(y)} R$, $Q \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} S$ *and* $z\#P$ *then* $P \mid Q \xrightarrow[rn]{\tau} new[z](R\{z|y\} \mid S)$*"*

Proof. We prove the direct implication $p \Rightarrow q$. Let us suppose that $P \xrightarrow[rn]{x(y)} R$, $Q \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} S$ and $z\#P$. Since $P \xrightarrow[rn]{x(y)} R$ and $z\#P$, by Theorem 5.7 and Remark 5.8 we have $P \xrightarrow[rn]{x(z)} R\{z|y\}$. Since we also have $Q \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} S$, from p we obtain $P \mid Q \xrightarrow[rn]{\tau} new[z](R\{z|y\} \mid S)$.

We prove now the converse implication $q \Rightarrow p$. Let us suppose that for some y we have $P \xrightarrow[rn]{x(y)} R$ and $Q \xrightarrow[rn]{y\, bn\, \overline{x}\langle y\rangle} S$. Since we work in the FM set theory, from the **finite support property** (the set of π-calculus terms form a nominal set) we can choose an atom z such that $z\#P, Q, R, S$. Since $Q \xrightarrow[rn]{y\, bn\, \overline{x}\langle y\rangle} S$ and $z\#Q$, by Theorem 5.6 and Remark 5.8 we have $Q \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} S\{z|y\}$. Now, because $z\#P$, we have that $P \mid Q \xrightarrow[rn]{\tau} new[z](R\{z|y\} \mid S\{z|y\}) = new[z]((R|S)\{z|y\})$. However $y\#(R|S)\{z|y\}$(the free occurrences of y in $R|S$ have already been removed when we applied the substitution $\{z|y\}$ to the process $R|S$). Hence $new[z]((R|S)\{z|y\}) = new[y]((R|S)\{z|y\}\{y|z\})$. However $z\#(R|S)$ which means $(R|S)\{z|y\}\{y|z\} = R|S$. We obtain $P \mid Q \xrightarrow[rn]{\tau} new[y](R \mid S)$. □

Analogue we can prove the following result:

Lemma 6.2. *In the restricted nominal semantics of the π-calculus the following sentences are equivalent:*

***p'**: "If $P \xrightarrow[rn]{x(z)} P'$ and $P \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} P''$ then $!P \xrightarrow[rn]{\tau} (new[z](P' \mid P''))|!P$'*

***q'**: "If $P \xrightarrow[rn]{x(y)} P'$, $Q \xrightarrow[rn]{z\, bn\, \overline{x}\langle z\rangle} P''$ and $z\#P$ then $!P \xrightarrow[rn]{\tau} (new[z](P'\{z|y\} \mid P''))|!P$"*

Lemma 6.3. *The trivial nominal semantics of the π-calculus and the nominal semantics of the π-calculus provide the same transitions.*

Proof. The trivial nominal semantics of the π-calculus is formed from the restricted nominal semantics of the π-calculus by adding the rules 11' and 14'. The nominal semantics of the π-calculus is formed from the restricted nominal semantics of the π-calculus by adding the rules 11 and 14. Actually, according to Remark 5.8 and to Lemmas 4.13, 4.14, the assertions p, p' and q, q' in the previous lemmas are the statement of the transition rules 11',14' and 11,14 respectively. So the trivial nominal semantics of the π-calculus is formed from the restricted nominal semantics of the π-calculus by considering the assertions p in Lemma 6.1 and p' in Lemma 6.2 as transition rules. The nominal semantics of the π-calculus is formed from the restricted nominal semantics of the π-calculus by considering the assertions q in Lemma 6.1 and q' in Lemma 6.2 as transition rules. The requested result follows from Lemmas 6.1 and 6.2. □

Theorem 6.4. *For any two processes P and Q and any action α, we have the following equivalence:*

$$P \xrightarrow[l]{u} Q \text{ if and only if } P \xrightarrow[n]{u} Q.$$

Proof. From Theorem 4.18 we know that $P \xrightarrow[l]{u} Q$ if and only if $P \xrightarrow[tn]{u} Q$. From Lemma 6.3 we know that $P \xrightarrow[tn]{u} Q$ if and only if $P \xrightarrow[n]{u} Q$. We obtain that $P \xrightarrow[l]{u} Q$ if and only if $P \xrightarrow[n]{u} Q$. □

7. Conclusion and Related Work

Using the nominal techniques presented in the second section of this paper, we provide the nominal semantics of the π-calculus. We get a compact representation of the transition rules which uses the quantifier $\forall$ and the *nominal quantifier* $\mathsf{И}$. This compact representation is obtained by using the finite support property, Proposition 2.23, Remark 2.24, and other technical results presented in the second part of the paper. Rule number 11 of the nominal semantics, allows us to express the π-calculus mobility in another form than in the other known semantics of the π-calculus. We can say that the nominal semantics is able to express the mobility of the π-calculus by using a transition rule which is assumed to be valid only in a more restrictive hypothesis than its related rules in the late semantics of the π-calculus.

We also make a comparison between the new nominal semantics and the late semantics of the π-calculus. We can define a semantics of the π-calculus, with transition rules expressed in a nominal form (using the nominal quantifier), which is **equivalent** with the late semantics of the π-calculus. Using similar techniques with those presented in [1], we prove that the trivial nominal semantics and the late semantics of the π-calculus have the same expressive power (Theorem 4.18).

In this paper we do not want just to rewrite the transition rules of the late semantics of the π-calculus in a compact nominal form in the same way we did in [1] with the transition rules of the Sangiorgi semantics of the πI calculus. We want to use the nominal quantifier to express the transition rules of the π-calculus in the "weakest" form we need for describing the mobility. We make a refinement of the transition rule $L\text{-}CLOSE_L$ such that we assume a similar rule to be valid only for those z which are fresh for P (and not for all z as in the late semantics of the π-calculus) because for describing the mobility this is weakest condition requested in the rule's hypothesis (Remark 4.23). The freshness condition requested in the hypothesis of the rule number 11 is motivated by the manner in which the links are moved in a virtual space of linked processes (Proposition 4.21). The nominal quantifier helps us to "encode" the condition "z fresh for P" in the hypothesis of the rule number 11 such that this rule is presented in a compact form.

Since the nominal semantics provides a complete description of the π-calculus, we can explain (using induction on the depth of the deduction tree) the behavior of the binding operators in π-calculus; we are referring to the both freshness operator *new* and the operator for substitution *input*. This approach allows us to prove Theorem 5.4.

Several properties of the abstractions *new* and *input* are also presented in a nominal form (see for example Theorem 5.6 and Theorem 5.7). These properties together with the FM axioms allow us to establish an equivalence result between the late semantics of the π-calculus and the nominal semantics of the π-calculus. The nominal semantics and the late semantics of the π-calculus have the same expressive power (Theorem 6.4) even if the rules number 11 and 14 in the nominal semantics of the π-calculus are assumed to be valid only when several freshness conditions are satisfied whilst the related rules $L\text{-}CLOSE_L$ and $L\text{-}REP_{close}$ in the late semantics of the π-calculus are assumed to be valid without any freshness conditions.

There exists many papers where process calculi are modeled by using different techniques for binding. In [7], D. Hirschkoff formalized a subset of the π-calculus excluding match, mismatch and sum in Coq by using de Bruijn indices. Higher order abstract syntax (HOAS) was used to model the π-calculus in both Isabelle [11] and in Coq [8]. However, approaches based on HOAS can suffer by some problems. For example the function spaces can be too large. Also, function spaces can destroy the inductive structure. When using HOAS terms, binders are represented as functions of type **name** $\rightarrow$ **term**. If these functions range over the entire function space they may produce exotic terms, so the formalizations have to ensure that those terms are avoided. The nominal logic is a first order logic and, hence, exotic terms can not appear. Moreover, in the nominal framework the existence of fresh names is axiomatically assumed (each time when a such a name is requested) whilst in HOAS, if we need to generate names, we may need to index all predicates and relations by an explicit set of known names (see [8]).

Nominal techniques have also been used in order to formalize the π-calculus. In [2] the π-calculus is formalized in Isabelle using the nominal datatype package [15]. In paper [4], M.Gabbay also uses nominal techniques to obtain new transition rules for the π-calculus. The new rules provide a formal presentation of "freshness π-calculus". The notion of nominal α-abstraction is used in [4] to present both *new* and *input* as a general binding operator; in this way this approach overlooks the important fact that in π-calculus there exist two different binding operators, and not only one like in λ-calculus. We study both the effect of *new* and the effect of substitution (induced by *input*) when they act together (Theorem 5.4), but we do not unify *new* and *input* by a single α-abstraction.

In our paper we present a nominal semantics of the π-calculus which is completely equivalent with the late semantics of the π-calculus. However our transition rules are *different* from those proposed by M. Gabbay. We want to emphasize the benefit of using nominal techniques in order to encode some freshness conditions which allow us to present the transition rules in the "weakest" form we need for characterizing the mobility mechanism. We show that we can characterize mobility by using a rule in the hypothesis of which we assume a freshness condition presented in the nominal approach. This rule is rule number 11 in Definition 4.5, which is assumed to be valid only when some freshness conditions are satisfied whilst its related $L\text{-}CLOSE_L$ in the late semantics of the π-calculus is assumed to be valid without any freshness conditions. The manner in which the links are moved in a virtual space of linked processes justify our approach for the rule number 11.

We have to recognize that the first idea of using nominal techniques to formalize the π-calculus belongs to M. Gabbay. In our paper we try to introduce a slightly different perspective (we do not say we introduce a better one!). Our nominal transition rules also provide a semantics of the π-calculus which is equivalent with the late semantics of the π-calculus. However our approach is motivated by the description of the mobility mechanism and our transition rules are different than those in [4].

The contribution of this paper is given mainly by the following aspects:

- We rewrite the transition rules of the late semantics of the π-calculus in a compact nominal form, using some techniques developed before in [1]. We use the nominal quantifier N to encode some freshness conditions in the hypothesis of the transition rules. We obtain the so called **trivial nominal semantics of the π-calculus** which has the same expressive power as the late semantics of the π-calculus (Theorem 4.18).

- We make a refinement of the rule $L\text{-}CLOSE_L$ in the late semantics of the π-calculus such that we rewrite this rule in the "weakest" form we need for expressing the links mobility. We obtain the so called **nominal semantics of the π-calculus**. The mobility mechanism can be described using the rule number 11 in the nominal semantics of the π-calculus (Figure 2) which is assumed to be valid only in a more restrictive hypothesis than its related $L\text{-}CLOSE_L$ in the late semantics of the π-calculus (Remark 4.23). The nominal quantifier helps us to encode the premise "z fresh for P" in the hypothesis of the rule number 11 such that this rule is presented compactly.

- We present several nominal properties of the abstractions **new** and **input** in the nominal semantics of the π-calculus (Proposition 5.1, Theorem 5.4, Theorem 5.6, Theorem 5.7). It is obvious to remark that these properties are also valid in the trivial nominal semantics of the π-calculus and in the restricted nominal semantics of the π-calculus (Remark 5.8).

- Using the nominal properties of the abstractions **new** and **input** and the finite support axiom in the FM set theory, we are able to prove that the nominal semantics and the late semantics of the π-calculus are completely equivalent. The result which states the equivalence between the nominal semantics and the late semantics of the π-calculus follows even if the rules number 11 and 14 in the nominal semantics of the π-calculus are assumed to be valid only when several freshness conditions are satisfied whilst the related rules $L\text{-}CLOSE_L$ and $L\text{-}REP_{close}$ in the late semantics of the π-calculus are assumed to be valid without any freshness conditions.

Acknowledgements. We thank the anonymous reviewers for helpful comments. This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-ID-PCE-2011-3-0919.

References

[1] Alexandru A., Ciobanu G., *Nominal semantics of the πI-calculus*, *Proceedings of the 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2011)*, IEEE Computer Society Press, pp. 331–339, 2012.

[2] Bengtson J., Parrow J., *Formalising the π-calculus using nominal logic*, Logical Methods in Computer Science, vol. **5**, pp. 1–36, 2009.

[3] Ciobanu G., Rotaru M., *A π-calculus machine*, Journal of Universal Computer Science, vol. **6**, pp. 39–59, 2000.

[4] Gabbay M., *The pi-calculus in FM*, Thirty Five Years of Automating Mathematics, Kluwer Applied Logic, vol. **28**, pp. 247–269, Kluwer, 2003.

[5] Gabbay M. J., Gabbay M., *Substitution for Fraenkel-Mostowski foundations*, *Proceedings of the 2008 AISB Symposium on Computing and Philosophy*, pp. 65–72, 2008.

[6] Gabbay M. J., Pitts A., *A new approach to abstract syntax with variable binding*, Formal Aspects of Computing, vol. **13**, pp. 341–363, 2002.

[7] Hirschkoff D., *A full formalisation of π-calculus theory in the calculus of constructions*, *Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics*, pp. 153–169, London, UK, 1997, Springer-Verlag.

[8] Honsell F., Miculan M., Scagnetto I., *π-calculus in (co)inductive type theory*, Theoretical Computer Science, **253**(2), pp. 239–285, 2001.

[9] Milner R., *Communicating and Mobile Systems: the π-Calculus*, Cambridge University Press, 1999.

[10] Pitts A., *Nominal logic, a first order theory of names and binding*, Information and Computation, **186**, pp. 165–193, 2003.

[11] Rockl C., Hirschkoff D., *A fully adequate shallow embedding of the π-calculus in Isabelle/HOL with mechanized syntax analysis*, J. Funct. Program., **13**(2), pp. 415–451, 2003.

[12] Rose J. S., *A Course on Group Theory*, Dover Publications, 1994.

[13] Sangiorgi D., Walker D., *The Pi-Calculus: A Theory of Mobile Processes*, Cambridge University Press, 2001.

[14] Sangiorgi D., *π-calculus, internal mobility, and agent-passing calculi*, Rapport INRIA nr. 2539, 1995.

[15] Urban C., *Nominal techniques in Isabelle/HOL*, Journal of Automated Reasoning, **40**(4), pp. 327–356, 2008.