ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **16**, Number 4, 2013, 261–286

Nominal Techniques for πI-Calculus

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 new semantics of the πI*-calculus*, namely the *nominal semantics*. A set of compact transition rules is given in terms of nominal logic by using a specific nominal quantifier. Based on several nominal techniques, it is proved an equivalence between the new nominal semantics and the original semantics of the πI-calculus, emphasizing the benefits of presenting the transition rules by using the nominal techniques.

1. Introduction

The aim of this paper is to present a set of *compact* transition rules for the πI-calculus. These transition rules are expressed using the quantifier $\forall$ and the nominal quantifier $\mathsf{И}$. Using some results presented in Section 2 we are able to prove that the new transition rules defined in this paper and the original transition rules presented in [21] provide the same transitions.

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 scopes, workspaces, or environments. Since renaming, binding and fresh names appear in several approaches, it became evident that they deserve to be studied in their own terms.

The FM set theory, also known as the theory of nominal sets, was originally developed in 1930s by Fraenkel and later by Mostowski to prove the independence of the Axiom of Choice from the other axioms of the Zermelo-Fraenkel (ZF) set theory. It was rediscovered and used by Gabbay and Pitts [13] to model the syntax of formal

systems involving variable binding operations. An advantage of modeling syntax in a model of the FM set theory is that datatypes of syntax modulo α-equivalence can be modeled inductively. This is because FM set theory delivers a model of variable symbols and α-abstraction. The FM axioms are precisely the Zermelo-Fraenkel with atoms (ZFA) axioms over an infinite set of atoms [13], together with the special property of finite support which claims that for each element x in an arbitrary set we can find a finite set of atoms supporting x. Since the support of x is defined as the least finite set of atoms supporting x, we can always find a fresh name for x, *i.e.*, an atom which is not in the support of x.

In λ-calculus, a general computation model, the α-equivalence class of a λ-term x have the support of x represented by the free variables with respect to x (Example 2.13). This means that Fraenkel-Mostowski set theory solves the problem of choosing fresh names, and it can be a more suitable framework for computer science.

The π-calculus was designed to be a foundation for the concurrent computation, in the same way as the λ-calculus is a foundation for sequential computation. The communication between processes in π-calculus is realized by some communication channels. Programs in the π-calculus are systems of parallel processes that synchronize via message-passing handshakes on named channels. The π-calculus embodies the view that in principle most, if not all, distributed computation may usefully be explained in terms of *exchanges of names on named communication channels*. The finite support property give us a mathematical reason for what such a renaming is always possible. Also, the FM set theory is the axiomatic support for the construction of the nominal logic.

Since we can use the same axioms of set theory for describing both the nominal logic and the π-calculus, we can try to apply the nominal techniques in the π-calculus. This is the central idea of [3]. The πI-calculus (or simply πI) was introduced by Sangiorgi in [21]; it is a variant of the π-calculus in which the free outputs are disallowed. We apply the nominal techniques to obtain a new semantics of the πI-calculus, in a similar way we did for the π-calculus in [3]. For this we use a special nominal quantifier $\mathsf{И}$ which provides the possibility of removing the free variables (which are represented by the notion of support) from the scope of a rule. $\mathsf{И}x$ means that x can be fresh for other parts of an expression; for example, $\forall x.\mathsf{И}y.\forall z.expression$ is true iff $\forall x, y, z.(y\, is\, fresh\, for\, x) \Rightarrow (expression)$ (more details are given in Proposition 2.18 and Proposition 2.19). A mixing of $\forall$ and $\mathsf{И}$ is used to replace the side conditions in the transition rules of this process calculus. Finally, we prove an equivalence between the expressive power of nominal semantics and the usual semantics of the πI-calculus presented in [21].

2. Fraenkel-Mostowski Set Theory

The aim of this section is to keep the paper self-contained by providing several nominal techniques used in order to define the nominal semantics of the πI-calculus. The notions we present are slightly modified versions of those introduced in [11, 13].

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 permutations of A which leave unchanged all but finitely many atoms. It can be proved that S_A is the group of all bijections $\pi : A \to A$ generated by composing finitely many transpositions.

Definition 2.2.

- Let X be a ZF-set. 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 ZF-set, 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 if 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 in 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. ([13]) *Let X be an S_A-set, and for each $x \in X$ let $\mathcal{F}_x = \{S \subset A \mid S$ finite, S supports $x\}$. If $\mathcal{F}_x$ is nonempty (particularly if X is a nominal set) then it has a least element which also supports x. We call this element* the support of x, *and we denote it by $supp(x)$.*

Proposition 2.6. ([13]) *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 $supp(\pi \cdot x) = \pi(supp(x))$.*

Example 2.7.

1. The set A of atoms is an S_A-set with the S_A-action $\cdot : S_A \times A \to A$ 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. Moreover, $supp(a) = \{a\}$ for each $a \in A$.

2. The set A of atoms is an S_A-set with the S_A-action $\cdot : S_A \times A \to A$ 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. Moreover, $supp(a) = \emptyset$ 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 $\{a \in A \,|\, \sigma(a) \neq a\}$ supports σ. Moreover, $supp(\sigma) = \{a \in A \,|\, \sigma(a) \neq a\}$ for each $\sigma \in S_A$.

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. Moreover, $supp(x) = \emptyset$ for each $x \in X$.

5. If $(X, \cdot)$ is an S_A-set then $\wp(X) = \{Y \,|\, Y \subseteq X\}$ is also an S_A-set with the S_A-action $\star : S_A \times \wp(X) \to \wp(X)$ defined by $\pi \star Y := \{\pi \cdot y \,|\, y \in Y\}$ for all permutations π of A, and all subsets Y of X. 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 the 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 in fact the action $\star$ restricted to $\wp_{fs}(X)$) is indeed included in $\wp_{fs}(X)$ because of Proposition .

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 $X + Y = \{(0, x) \,|\, x \in X\} \cup \{(1, y) \,|\, y \in Y\}$. $X + Y$ 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.

As in [13] 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) $FM(A)$ whose all elements have the finite support property. One benefit is that if a particular construction can be expressed in this language, 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 the usual von Neumann cumulative hierarchy of sets:

- $\nu_0 = \emptyset$;

- $\nu_{\alpha+1} = \wp(\nu_\alpha)$;
- $\nu_\lambda = \underset{\alpha\leq\lambda}{\cup} \nu_\alpha$ (λ a limit ordinal).

More generally, given a set U we can define analogue a cumulative hierarchy of sets involving atoms from U [13]:

- $\nu_0(U) = \emptyset$;
- $\nu_{\alpha+1}(U) = U + \wp(\nu_\alpha(U))$;
- $\nu_\lambda(U) = \underset{\alpha\leq\lambda}{\cup} \nu_\alpha(U)$ (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 in Example 2.7):

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

From Example 2.7 each $FM_\alpha(A)$ is a nominal set. When we consider the union of all $FM_\alpha(A)$ we get 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 is denoted by $FM(A)$. Using the names *atm* and *set* for the functions $x \mapsto (0, x)$ and $x \mapsto (1, x)$ (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 elements of the form $set(X)$ CHFM-sets and the elements of the form $atm(a)$ atoms.

The 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 if and only if 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 $S_A \times X$ will have the

codomain equal with X because $\pi \cdot X = X$ for all $\pi \in S_A$. Nominal CHFM-sets will be S_A-sets and nominal sets as well.

We provide an axiomatic presentation of the Fraenkel-Mostowski (FM) set theory. The axioms are the ZFA axioms (see [13]) together with the finite support property (axiom 11). We convene that the symbol "$\Rightarrow$" means "implies", and the symbol "$\Leftrightarrow$" represents "if and only if".

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 clear 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 an 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 will be called *interchange function*.

Definition 2.8.

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 the ZFA set theory 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 $\pi \cdot x := \{\pi \cdot y \,|\, y \in x\}$, 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 the axioms of the ZFA set theory, and $\cdot : S_A \times X \to X$ is an interchange function on X.

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

Remark 2.9. *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 2.10. 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 2.8 always allows one to find a finite set supporting x, for each element x in an arbitrary NFM-set.

The following theorem follows immediately from Theorem 2.5.

Theorem 2.11. *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 we denote it by $supp(x)$.*

Example 2.12. We present some examples of how we can calculate 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.

The following Example (considered also in [13]) shows us how we can express the usual λ-calculus in FM.

Example 2.13.

1. If X' is the set of λ-terms t, then we inductively 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 for 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 [13]). 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 (see [13]). 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 [13]. α-equivalent terms are identified in the nominal framework since two α-equivalent terms have the same set of free variables.

Definition 2.14. If $x \in A$ and $y \in Y$ where Y is a nominal set, we say that x *is fresh for* 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 $\forall x.\exists a \in A.a\#x$.

The following equivariance property hold in FM. A complete proof of this property can be found in [13].

Remark 2.15. *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.*

Definition 2.16. Let P be a predicate on A. We say that $\mathsf{M}a.P(a)$ if $P(a)$ is true for all but finitely many elements of A. M is called the **nominal quantifier.**

The proof of the next result is simple, and uses only the definition of support [10].

Proposition 2.17.

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 $supp(X) = \cup\{supp(x) \mid x \in X\}$.

Proposition 2.18. *Let $(x_i)_i$ be a set of distinct variables and p a formula in the logic of FM. We have the following implications: $[\forall a \in A.(a\#(x_i)_i \Rightarrow p)] \Rightarrow [И a.p] \Rightarrow [\exists a \in A.(a\#(x_i)_i \wedge p)]$.*

Proof: We know that the set $\{a \in A \mid a\#(x_i)_i\}$ is cofinite. If we assume 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, $И a.p$. Now, if we assume $И a.p$, 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 number 10. □

Proposition 2.19. *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 $[\forall a \in A.(a\#(x_i)_i \Rightarrow p)]$.*

Proof: Let us suppose that for some $a \in A$ we have $a\#(x_i)_i \wedge p$. Then, by Remark 2.15, 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 2.20. Let X be a nominal set and $u \in X$. If $B \subset A$, we define $u||B \stackrel{def}{=} \{\pi \cdot u \mid \pi \in Fix(B)\}$; $u||B$ is called *the freshness orbit of u on B*.

Remark 2.21. *$u||B$ 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 $a||\{b\} = \{a, c, d, e, ...\}$.*

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

Definition 2.22. 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 $[a]x = \cap\{V \subset A \times X \mid (a, x) \in V \wedge supp(V) \subset supp(x) \setminus \{a\}\}$.*
2. *We define the* abstraction function *to be the function $abs : A \times X \to [A]X = \{[a]x \mid a \in A \wedge x \in X\}$, $(a, x) \to [a]x$.*

We often use the notion of α-abstraction for abstractive elements by analogy with the abstraction in the λ-calculus. Corollary 2.24 and Example 2.13 make this analogy possible. We can prove that $[a]x = (a, x)||(supp(x) \setminus \{a\})$.

Theorem 2.23. ([11]) *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 $supp(x) \setminus \{a\}$.*

Corollary 2.24. ([13]) *Let X be a nominal set, $a \in A$ and $x \in X$. Then we have*
a) $[a]x = \{(y,(y,a)x) \mid y \in A, y \neq a \text{ and } y\#x\} \cup \{(a,x)\}$.
b) $supp([a]x) = supp(x) \setminus \{a\}$.

Example 2.25.

- $[a](A\backslash\{a\}) = \{(a, A\backslash\{a\}), (b, A\backslash\{b\}), (c, A\backslash\{c\}), ...\}$
- $[a]\{a,b\} = \{(a,\{a,b\}), (c,\{c,b\}), (d,\{d,b\}), (e,\{e,b\}), ...$

From [10] we know the following result:

Proposition 2.26. *Let X be a nominal set. Let $a, b \in A$ 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 2.27. *Let X be a nominal set. Let $a, b \in A$ 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 2.28. *For a λ-term x, we denote by $fn(x)$ the free names for x. We can say (according to Corollary 2.24 and Example 2.13) that $fn([a]x) = fn(x) \setminus \{a\}$. This means that* [ ] can be seen as a binding of a in x.

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 2.22. The element $[a]x$ will be 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 number 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 2.26 and Corollary 2.27 are valid when X is $FM(A)$, that is, x, y are FM-sets. More details are in [13].

3. πI-Calculus

The π-calculus [17] is a widely accepted model of interacting systems with dynamically evolving communication topology; π-calculus allows channels to be passed as data along other channels, and this introduces a channel mobility. We work with the monadic version of the π-calculus. The **processes** are defined over the set $\mathcal{X}$ of names by the following grammar:

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

0 is the empty process. 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 now they share the same channel name. The 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. 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 will replace 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 3.1. 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, or it lies 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 will be 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 $new\, zP$, that means that the components of P can use z to interact with one another but not with other processes. Also, (in $\pi-calculus$) in $new\, zP$, 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 3.2. (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 (and denote it by $P \equiv_\alpha Q$)* if Q can be obtained from P by a finite number of changes of bound names.

It is easy to remark that α-convertibility in π- calculus is the same as the α-equivalence in λ-calculus. In [22] the processes which are α-convertible are identified by *convention*. In the nominal approach we do don't need to use any convention. We shall *prove* that two processes are α-convertible if and only if they are equal. In the FM-approach we are able to prove mathematically (as in Example 2.13) that α-convertible processes are identical, whilst in the ZF approach we assume this fact to be valid only by convention.

A complete presentation of the π-calculus is given in [17] and [22].

In [21], Sangiorgi separates the mobility mechanisms of the π-calculus into two, respectively called *internal mobility* and *external mobility*. The former arises when an input meets a bound output, *i.e.*, the output of a private name; the latter arises when an input meets a free output, i.e., the output of a known name. Internal mobility is responsible for much of the expressiveness of the π-calculus, whereas external mobility is responsible for much of the semantic complications.

The main distinction between the internal and external mobility is clearly explained in Section 2.2 of [21]. The internal mobility appears when a bound output interacts with an input: $\overline{x}(z).P \,|\, x(z).Q \xrightarrow{\tau} new\, z(P|Q)$. The interaction consumes the two prefixes but leave unchanged the derivatives underneath. With internal mobility, α-conversion is the only form of name substitution involved. The external mobility appears when a free output interacts with an input: $\overline{x}\langle y\rangle.P \,|\, x(z).Q \xrightarrow{\tau} P|Q\{y|z\}$. In this case a substitution must be imposed. We present the πI-calculus following [21].

Definition 3.3. The class of (finite) πI processes is described by the following grammar:

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

In πI, the input and output constructs are truly symmetric: Since only outputs of private names are possible, an input $x(z).P$ means "receive a fresh name at x", which is precisely the dual of the output $\overline{x}(z).P$. Indeed, we can define a "*dual*" operation which transforms every output into an input and vice versa: the symmetry of the calculus is then manifested by the fact that *dual* commutes with the transition relation [21]. The transition rules are labelled by the actions. In πI, we have three kinds of actions:

Definition 3.4. The *actions* in the πI-calculus are given by

$$\alpha ::= \overline{x}(z) \mid x(y) \mid \tau.$$

The *bound output* action $\overline{x}(z)$ is sending a fresh name z via x; here x is a free name, and z is a bound name. The *bound input* action $x(y)$ is receiving a fresh name y via x; here x is a free name, and y is a bound name. τ is the internal action; no free and bound names.

The transition relation labelled by α is written, in πI, $\xrightarrow[\pi I]{\alpha}$ where α symbolizes the action, and πI indicates the used calculus. For example $P \xrightarrow[\pi I]{\overline{x}(y)} Q$ means that P can

send y via x and evolve to Q; $P \xrightarrow[\pi I]{x(y)} Q$ means that P can receive y via x and evolve to Q.

Definition 3.5. The following labeled transition rules define *the Sangiorgi semantics of the πI- calculus.*

$$OUT: \quad \frac{}{\overline{x}(y).P \xrightarrow[\pi I]{\overline{x}(y)} P} \qquad INP: \quad \frac{}{x(y).P \xrightarrow[\pi I]{x(y)} P}$$

$$TAU: \quad \frac{}{\tau.P \xrightarrow[\pi I]{\tau} P} \qquad SUM: \quad \frac{P \xrightarrow[\pi I]{\alpha} P'}{P + Q \xrightarrow[\pi I]{\alpha} P'}$$

$$ALPHA: \quad \frac{P \equiv_\alpha Q \,,\, Q \xrightarrow[\pi I]{\alpha} R}{P \xrightarrow[\pi I]{\alpha} R},$$

where $P \equiv_\alpha Q$ means that Q can be obtained from P after an α-conversion.

$$PAR: \quad \frac{P \xrightarrow[\pi I]{\alpha} P'}{P \mid Q \xrightarrow[\pi I]{\alpha} P' \mid Q}, \quad bn(\alpha) \cap fn(Q) = \emptyset$$

$$COM: \quad \frac{P \xrightarrow[\pi I]{\overline{x}(z)} P' \,,\, Q \xrightarrow[\pi I]{x(z)} Q'}{P \mid Q \xrightarrow[\pi I]{\tau} new\, z(P' \mid Q')} \qquad RES: \quad \frac{P \xrightarrow[\pi I]{\alpha} P'}{new\, zP \xrightarrow[\pi I]{\alpha} new\, zP'}, \; z \notin n(\alpha)$$

4. Nominal Semantics of the πI-Calculus

We rewrite the transition rules from Definition 3.5 in a nominal form by using specific nominal techniques. In the same way we did for the π-calculus in [3], we can express the transition rules in πI in a compact form.

Using the same arguments as in Example 2.13, we can say that the set of the πI-calculus terms form an NFM-set and the set of the πI-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 πI-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 item 1), we can define the abstraction between a variable and a πI-calculus term in the sense of Definition 2.22. In each of the expressions $\overline{x}(y).P$, $x(y).P$ and $new\, yP$, we have that y is bound. We can use the notion of abstraction and we write these expressions like $\overline{x}[y]P$, $x[y]P$ and $new[y]P$, respectively. In this case we can see bindings represented by *new* and *input* as α-abstractions (or abstractive elements of Definition 2.22) 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 (each time) as an α-abstraction in the sense of Definition 2.22. However α-abstraction is used to emphasize that, when we apply new zP (or $x(z).P$ or $\overline{x}(z).P$), 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-abstraction generated by new and input, ***bindings****.*

In the nominal approach we do not define a structural congruence. However remark that processes which are obtained from one another by an α-conversion are identified in the nominal framework (Example 2.13). Before defining the transition rules of the new (nominal) semantics of the πI-calculus we state the following property:

Remark 4.2. *The syntax and the basic actions: bound input, bound output and internal action are the same both in the Sangiorgi semantics and in the nominal semantics of the πI-calculus with the mention that α-equivalent terms are identified in the nominal approach.*

We make the distinction between the global binding (given by quantifiers) and local binding (given by *output*, *input* and *new*). If a variable appears bound one time locally and second time globally in the same transition rule, this thing does not represent a logical contradiction. In the following rules, the binding provided by *new* and *input* (in the sense of Remark 4.1) will be seen as a local binding only for the following process (for example in the expressions $\overline{x}[y]P$, $x[y]P$ and $new[y]P$, y binds the free occurrences of y only in P), and the binding made by the quantifiers $\forall$ and $\mathsf{И}$ will be seen as a global binding, for the entire rule. So, it is possible to have, in some rules, expressions of forms $\forall y, ...f_1(x[y]P, ...)$, $\forall y, ...f_2(new[y]P, ...)$, $\forall y, ...f_3(\overline{x}[y]P, ...)$, $\mathsf{И}y, ...f_4(x[y]P, ...)$, $\mathsf{И}y, ...f_5(new[y]P, ...)$ or $\mathsf{И}y, ...f_6(\overline{x}[y]P, ...)$. The formulas $\forall y, ...f_1(x[y]P, ...)$, $\forall y, ...f_2(new[y]P, ...)$, and $\forall y, ...f_3(\overline{x}[y]P, ...)$ express that $f_1(x[y]P, ...)$, $f_2(new[y]P, ...)$, and respectively $f_3(\overline{x}[y]P, ...)$ are valid, and their validity does not depend of y. Also, $\mathsf{И}y, ...f_4(x[y]P, ...)$, $\mathsf{И}y, ...f_5(new[y]P, ...)$ and $\mathsf{И}y, ...f_6(\overline{x}[y]P, ...)$ means that $f_4(x[y]P, ...)$, $f_5(new[y]P, ...)$, and respectively $f_6(\overline{x}[y]P, ...)$ 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

$$\forall x, y.\mathsf{И}z.\forall P, Q.\frac{P \xrightarrow[nI]{x(y)} Q}{new[z]P \xrightarrow[nI]{x(y)} new[z]Q},$$

where by $\cdot \xrightarrow[nI]{} \cdot$ we denote a transition in the nominal semantics of the πI, is correctly written even we should be tempted to say that y and z are bound two times. The variables x, y and z are globally bound one time. The second binding, made by *new* and *input*, is a local one, and it is restricted to the processes P and Q (or respectively to the action $x(y)$). So, to say that

$$\forall x, y.\mathsf{И}z.\forall P, Q.\frac{P \xrightarrow[nI]{x(y)} Q}{new[z]P \xrightarrow[nI]{x(y)} new[z]Q}$$

is the same thing with saying that

$$P \xrightarrow[nI]{x(y)} Q \text{ implies } new[z]P \xrightarrow[nI]{x(y)} new[z]Q$$

if z satisfy some cofinitess conditions (we'll prove later that these conditions are: z is fresh both for x and for y).

Definition 4.3. The following labeled transition rules define the **nominal semantics** of the πI-calculus.

1. $\forall x, y, P. \overline{x}[y]P \xrightarrow[nI]{\overline{x}(y)} P$ *(n-bound output)*

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

3. $\forall P. \tau.P \xrightarrow[nI]{\tau} P$ *(n-tau)*

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

5. $\dfrac{P =_\alpha Q \ , \ Q \xrightarrow[nI]{\alpha} R}{P \xrightarrow[nI]{\alpha} R}$,
   where $P =_\alpha Q$ means that $P = Q$ and Q can be obtained from P by a renaming of bound names in P

6. For parallel composition we have the following three transition rules (for bound output, bound input and τ):

   (a) $\forall R. \curlywedge y. \forall x, P, Q. \dfrac{P \xrightarrow[nI]{\overline{x}(y)} Q}{P|R \xrightarrow[nI]{\overline{x}(y)} Q|R}$

   (b) $\forall R. \curlywedge y. \forall x, P, Q. \dfrac{P \xrightarrow[nI]{x(y)} Q}{P|R \xrightarrow[nI]{x(y)} Q|R}$

   (c) $\forall P, Q, R. \dfrac{P \xrightarrow[nI]{\tau} Q}{P|R \xrightarrow[nI]{\tau} Q|R}$

7. For restriction we have the following three transition rules (for bound output, bound input and τ):

(a) $\forall x, y. \text{И} z. \forall P, Q. \dfrac{P \xrightarrow[nI]{\overline{x}(y)} Q}{new[z]P \xrightarrow[nI]{\overline{x}(y)} new[z]Q}$

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

(c) $\forall P, Q, z. \dfrac{P \xrightarrow[nI]{\tau} Q}{new[z]P \xrightarrow[nI]{\tau} new[z]Q}$

8. $\forall P, Q, x, z, R, S. \dfrac{P \xrightarrow[nI]{\overline{x}(z)} R \,,\; Q \xrightarrow[nI]{x(z)} S}{P|Q \xrightarrow[nI]{\tau} new[z](R|S)}$ *(n-com)*

Remark 4.4. *The presence of the symbol* $=_\alpha$ *in the rule number 5 in Definition 4.3 is justified by the fact that, in the nominal framework, the relationship* $\equiv_\alpha$ *does not have a special meaning. The* α*-equivalent terms are identified in the nominal framework (see Example 2.13). Since the* α*-convertibility in* π*-calculus is the same as the* α*-equivalence in the* λ*-calculus,* $P \equiv_\alpha Q$ *means that* Q *can be obtained from* P *after an* α*-conversion, and in the nominal approach means that* $P = Q$*. We denote this equality by* $=_\alpha$*. According to Proposition 2.26, we can obtain the following result:*

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* $\overline{a}[x]P = \overline{b}[y]Q$*, then* $a = b$ *and one of the following sentences is true:*
   - $x = y$ *and* $P = Q$*,*
   - $x \neq y$ *and* $x \# Q$ *and* $P = (x\, y) \cdot Q$*.*
3. *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* $P = (x\, y) \cdot Q$*.*

Since every process is built by a recursive scheme as in Definition 3.3, a straightforward inductive method on the structure of the processes and the result presented in this remark shows us that, if $P = Q$ *in the nominal framework, then* Q *is obtained from* P *after a renaming of several bound names.*

In this way, rule number 5 is trivial and could be eliminated. *However, we chose to present it in the nominal semantics of the* πI*-calculus only to emphasize that the*

rule ALPHA in the Sangiorgi semantics of the πI also have an analogue (the rule number 5) in the nominal semantics of the πI.

In a transition rule, the free name are assumed (by convention) to be different from the bound names. This assumption is possible since of the finite support property of the FM set theory.

When, for a transition rule of form $\forall x.И y.\forall P,Q.\frac{P\xrightarrow[nI]{u(x,y,\ldots)}Q}{R\xrightarrow[nI]{v(x,y,\ldots)}S}$ (1) (where $u(x,y,...)$ and $v(x,y,...)$ are not input actions; they are actions depending on x or on y...), we have an analogue in the Sangiorgi semantics of the πI-calculus of form $\frac{P\xrightarrow[\pi I]{u(x,y,\ldots)}Q}{R\xrightarrow[\pi I]{v(x,y,\ldots)}S}$, whenever $y \neq x$(2) (when this analogy is possible in the sense of Proposition 2.18 and Proposition 2.19), we say that (1) and (2) are identical by convention (they have the same effect) even they are different expressions. However, in this case, we prefer to use nI in (1) only for emphasizing that (1) is a nominal semantic rule where quantifier $И$ is present. Note that this identification (of (1) with (2)) is made by convention, since the basic actions in nominal semantics are identical with those in the Sangiorgi semantics of the πI-calculus.

The notion of *identical by convention* is not expressed mathematically; it is only intuitive. We give a formal interpretation of this notion, and give a mathematical proof of the equivalence of the Sangiorgi and of the nominal semantics of the πI. First we define a set of transition rules in the form that Sangiorgi did (assuming some cofinitess conditions in the rule's hypothesis). Next we prove that these rules are in fact identical with those in the nominal semantics of the πI.

Definition 4.5. A basic transition rule in the nominal semantics of πI is in a *Sangiorgi-nominal form* if it coincides with one of the following rules:

$$OUT_{snI}: \quad \frac{}{\overline{x}[y]P \xrightarrow[nI]{\overline{x}(y)} P} \qquad INP_{snI}: \quad \frac{}{x[y]P \xrightarrow[nI]{x(y)} P}$$

$$TAU_{snI}: \quad \frac{}{\tau.P \xrightarrow[nI]{\tau} P} \qquad SUM_{snI}: \quad \frac{P \xrightarrow[nI]{\alpha} P'}{P+Q \xrightarrow[nI]{\alpha} P'}$$

$$ALPHA_{snI}: \quad \frac{P =_\alpha Q \ , \ Q \xrightarrow[nI]{\alpha} R}{P \xrightarrow[nI]{\alpha} R},$$

where $P =_\alpha Q$ means that $P = Q$ *and* Q *can be obtained from* P *after a renaming of bound names in* P

$$PAR_{snI}: \quad \frac{P \xrightarrow[nI]{\alpha} P'}{P \mid Q \xrightarrow[nI]{\alpha} P' \mid Q}, \quad bn(\alpha) \cap supp(Q) = \emptyset$$

(bn is a convention of notation in the sense of Remark 4.6)

$$COM_{snI}: \quad \frac{P \xrightarrow[nI]{\overline{x}(z)} P' \ , \ Q \xrightarrow[nI]{x(z)} Q'}{P \mid Q \xrightarrow[nI]{\tau} new[z](P' \mid Q')}$$

$$RES_{snI}: \quad \frac{P \xrightarrow[nI]{\alpha} P'}{new[z]P \xrightarrow[nI]{\alpha} new[z]P'}, z \notin bn(\alpha) \, and \, z\#\alpha$$

(bn is a convention of notation cf. Remark 4.6).

We give some lemmas allowing us to prove that *each transition rule in the nominal semantics of the πI-calculus can be presented in a Sangiorgi-nominal form.* Later we prove a result saying that the Sangiorgi and the nominal of the πI provide the same transitions.

Remark 4.6. *We recall that a function as bn is not defined in the nominal logic. In the rules of Definition 4.5, $bn(\alpha)$ is only a notation for the names of α which are "bound" by new or input (in the sense of Remark 4.1). To eliminate bn from the rules of Definition 4.5, we can split each transition rule which has conditions where bn is present into two rules: one for transitions with bound names, and one for ones without bound names. In our case, we present only one rule where bn is only a* notation *with no 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 $bn(\alpha)$ in that rule; if α has no bound names, then $\emptyset$ is* denoted *by $bn(\alpha)$ in that rule.*

The convention of notation presented in Remark 4.6 is often used in the proof of the following lemmas.

Lemma 4.7.

1. *The rule 1 in the nominal semantics of the πI-calculus is identical with the rule OUT_{SnI}.*
2. *The rule 2 in the nominal semantics of the πI-calculus is identical with the rule INP_{SnI}.*
3. *The rule 3 in the nominal semantics of the πI-calculus is identical with the rule TAU_{SnI}.*
4. *The rule 4 in the nominal semantics of the πI-calculus is identical with the rule SUM_{SnI}.*
5. *The rule 5 in the nominal semantics of the πI-calculus is identical with the $ALPHA_{SnI}$.*
6. *The rule 8 in the nominal semantics of the πI-calculus is identical with the rule OUT_{SnI}.*

In the view of Proposition 2.18 and Proposition 2.19, we are now able to provide the following results:

Lemma 4.8. *The rule PAR_{SnI} of Definition 4.5 is identical with either the rule number 6 item (a) (if the related action α is a bound output), (b) (if the action α is a bound input) or (c) (if α is the internal action) in the nominal semantics of the πI-calculus.*

Proof: If α is the internal action, then $bn(\alpha) = \emptyset$ (bn represents only a notation and not the function bn in λ-calculus; see Remark 4.6) and the proof is trivial. Let us suppose now that α have a bound name. For example let y be the bound name of α and we chose an arbitrary Q. We study two cases: y can be bound by *output* or by *input*. Let p be the formula "$\forall x, P, P'.(P \xrightarrow[nI]{\overline{x}(y)} P'$ implies $P|Q \xrightarrow[nI]{\overline{x}(y)} P'|Q)$" (where y is bound by *output*) and q be the formula "$\forall x, P, P'.(P \xrightarrow[nI]{x(y)} P'$ implies $P|Q \xrightarrow[nI]{x(y)} P'|Q)$" (where y in bound by *input*). The free variable of p are contained in the set $\{y, Q\}$. Indeed, the set of free variables of p is formed only by Q (precisely only by the set of free names of Q, which is included in Q) because P, P', x are bound by the quantifier $\forall$ and y is bound by *output*; 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,Q, and again these are contained in the set $\{y, Q\}$. The free variables of q are contained in the set $\{y, Q\}$. 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', x are bound by the quantifier $\forall$ and y is bound by *input*; 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 y,Q, and again these are contained in the set $\{y, Q\}$. Now, according to Proposition 2.18 and Proposition 2.19, we obtain the equivalence result: "$\forall y.(y\#Q$ implies $p)$" if and only if "$\curlywedge y.p$". Also, the assertion $\forall y.(y\#Q$ implies $q)$ is equivalent with $\curlywedge y.q$. Since Q was chosen arbitrarily we can say that the rule PAR_{SnI} in the Definition 4.5 is exactly the rule number 6 in the nominal semantics of the $\pi I-$calculus (item (a) or (b) as y is bound by *output* respectively by *input*). □

Lemma 4.9. *The rule RES_{SnI} of Definition 4.5 is identical with either the rule number 7 item (a) (if the related action α is a bound output), (b) (if the action α is a bound input) or (c) (if α is the internal action) in the nominal semantics of the πI-calculus.*

Proof: If α is the internal action then $n(\alpha) = \emptyset$ and proof is trivial. Let us suppose now that α have a bound name. For example let y be the bound name of α. We study two cases: y can be bound by *output* or by *input*. Let p be the formula "$\forall P, P'.(P \xrightarrow[nI]{\overline{x}(y)} P'$ implies $new[z]P \xrightarrow[nI]{\overline{x}(y)} new[z]P')$" (where y is bound by *output*) and q be the formula "$\forall P, P'.(P \xrightarrow[nI]{x(y)} P'$ implies $new[z]P \xrightarrow[nI]{x(y)} new[z]P')$" (where y is bound by *input*). The free variables of p are contained in the set $\{x, y, z\}$ (the set of free variables of p is formed only by x because P, P' are bound by the quantifier $\forall$, y is bound by *output* 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 x, y, z, and again these are contained in the set $\{x, y, z\}$) and the free variables of q are

contained in the set $\{x, y, z\}$ (the set of free variables of q is formed only by x because P, P' are bound by the quantifier $\forall$, y is bound by *input* and z 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, y, z, and again these are contained in the set $\{x, y, z\}$). Now, according to Proposition 2.18 and Proposition 2.19, we obtain the equivalence result: "$\forall z.(z\#\{x, y\}$ implies $p)$" if and only if "$И z.p$" Also we have "$\forall z.(z\#\{x, y\}$ implies $q)$" if and only if "$И z.q$". Since x, y were chosen arbitrarily and because for the action α we have $n(\alpha) = \{y\} \cup fn(\alpha)$, i.e., $n(\alpha) = \{y\} \cup supp(\alpha)$ we can say that the rule RES_{SnI} in Definition 4.5 is exactly the rule number 7 in the nominal semantics of the $\pi I-$calculus (item (a) or (b) as y is bound by *output* respectively by *input*). □

Corollary 4.10. *The Sangiorgi-nominal transition rules of Definition 4.5 represents also the nominal semantics of the πI-calculus.*

Proof: Since each transition rule in the nominal semantics of the πI-calculus described in Definition 4.3 can be expressed in a Sangiorgi-nominal form (see Lemmas 4.7, 4.8, 4.9) we have that each transition rule presented in Definition 4.3 is identical with a related rule in Definition 4.5. This identification is made by an equivalence and not only by a convention. It is clear that the transition rules in Definition 4.5 also represents the nominal semantics of the πI-calculus. □

The main result of this paper is the following theorem.

Theorem 4.11. *For any two processes P and Q and any action α we have*

$$P \xrightarrow[\pi I]{\alpha} Q \text{ iff } P \xrightarrow[nI]{\alpha} Q.$$

Before starting the proof of Theorem 4.11 we remind how we can prove, in the general theory, the equivalence of two semantics of several calculi. For proving that two semantics of the πI-calculus, namely 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 an encoding (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 practical way to do this is to define the morphism φ inductively by the following 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 the prefixes in π_1 and each of the prefixes p_1^i have an homologue 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. $\varphi(new\, x P_1) = new\, x \varphi(P_1)$, 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 (or πI) we must show (the classical procedure is by induction after 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 homologue corespondent in π_2 of α_1 (for example if π_1 is the late semantics of the π-calculus and π_2 is the nominal semantics of the π-calculus described in [3] 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 homologue 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 (restricted) 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 πI-calculus, the things are very clear. Both in the Sangiorgi semantics of the πI-calculus and in the nominal semantics of the πI-calculus, *the processes are defined by the same syntax as in Definition 3.3 (up to a renaming of "bound" names) and the basic actions are the same* (the actions are presented in Definition 3.4). So, it is not necessary to define explicitly a morphism φ between the Sangiorgi semantics of the πI-calculus and the nominal semantics of the πI-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 framework) and whose kernel is the α-equivalence on the Sangiorgi semantics of the πI-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 proof 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 an *inclusion* for an easy writing).

In our mind it is possible to apply the rule $ALPHA$ in the πI when a choice of some fresh names is requested. However we do not specify, each time, explicitly when we replace a process with an α-equivalent process. We can start the proof of Theorem 4.11.

Proof: For the nominal semantics of the πI-calculus we consider the set of transition rules presented in Definition 4.5 instead of the transition rules presented in Definition 4.3. By Corollary 4.10, these two sets of transition rules coincide. Each transition of form $P \xrightarrow[nI]{\alpha} Q$ is obtained by applying repeatedly the basic transition rules in Definition 4.5. We have to prove a double implication. The first part will be proved by induction after the depth of the deduction tree of πI, the second by induction

after the depth of the deduction tree of nI. To save space we do not rewrite out the induction hypotheses in complete formality. We analyze only few cases. The rest of them are analogue or trivial. First we prove that $P \xrightarrow[\pi I]{\alpha} Q$ implies $P \xrightarrow[nI]{\alpha} Q$.

Case PAR. 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 $P \xrightarrow[\pi I]{x(y)} P'$ and $y \notin fn(Q)$ for some Q. Then $P \mid Q \xrightarrow[\pi I]{x(y)} P' \mid Q$. By the inductive hypothesis $P \xrightarrow[nI]{x(y)} P'$. Since $y \notin fn(Q)$ is the same with $y\#Q$ (precisely with $y\#\varphi(Q)$; however we made a convention to eliminate φ for an easy writing), we can apply the rule PAR_{SnI} and we obtain $P \mid Q \xrightarrow[nI]{x(y)} P' \mid Q$.

Case 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 $P \xrightarrow[\pi I]{\overline{x}(y)} P'$ and $z \notin n(\overline{x}(y))$. Then $new\, zP \xrightarrow[\pi I]{\overline{x}(y)} new\, zP'$. By the inductive hypothesis $P \xrightarrow[nI]{\overline{x}(y)} P'$. 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 4.6), we can apply the rule RES_{SnI} and we obtain $new[z]P \xrightarrow[nI]{\overline{x}(y)} new[z]P'$.

Case $ALPHA$. Suppose $Q \xrightarrow[\pi I]{\alpha} R$ and $P \equiv_\alpha Q$. Then $P \xrightarrow[\pi I]{\alpha} R'$. By the inductive hypothesis $Q \xrightarrow[nI]{\alpha} R$. Since $P \equiv_\alpha Q$ means in the FM approach that $P = Q$ (we denoted this by $P =_\alpha Q$) as it was proved in Example 2.13 or in Remark 4.4, it is now clear that $P \xrightarrow[nI]{\alpha} R$.

We prove the second implication: $P \xrightarrow[nI]{\alpha} Q$ implies $P \xrightarrow[\pi I]{\alpha} Q$.

Case PAR_{SnI}. 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 $P \xrightarrow[nI]{x(y)} P'$ and $y\#Q$ for some Q. Then $P \mid Q \xrightarrow[nI]{x(y)} P' \mid Q$ by PAR_{SnI}. By the inductive hypothesis $P \xrightarrow[\pi I]{x(y)} P'$. Since $y\#Q$ means $y \notin supp(Q) = fn(Q)$, we can apply the rule PAR and we obtain $P \mid Q \xrightarrow[\pi I]{x(y)} P' \mid Q$.

Case RES_{SnI}. 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 $P \xrightarrow[nI]{\overline{x}(y)} P'$ and $z\#\overline{x}(y)$, $z \notin bn(\overline{x}(y))$. Then $new[z]P \xrightarrow[nI]{\overline{x}(y)} new[z]P'$ by RES_{SnI}. By the inductive hypothesis $P \xrightarrow[\pi I]{\overline{x}(y)} P'$. Since $z\#\overline{x}(y)$ and $z \notin bn(\overline{x}(y))$ ($bn(\alpha)$ is only a notation in the sense of Remark 4.6) 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 RES and we obtain $new\, zP \xrightarrow[\pi I]{\overline{x}(y)} new\, zP'$.

Case $ALPHA_{SnI}$. Suppose $Q \xrightarrow[nI]{\alpha} R$ and $P = Q$. Then, clearly $P \xrightarrow[nI]{\alpha} R'$ (this thing is trivial; however we can also apply the *trivial* rule $ALPHA_{SnI}$). By the inductive hypothesis $Q \xrightarrow[\pi I]{\alpha} R$. Since $P = Q$ means that $P \equiv_\alpha Q$ (in the view of Remark 4.4), we can apply the rule $ALPHA$ in πI and we obtain that $P \xrightarrow[nI]{\alpha} R$. □

Remark 4.12. *If we apply step by step the method of proving the equivalence of several semantics presented before, in the proof of the Theorem 4.11 we obtain (by induction) that* $P \xrightarrow[nI]{\alpha} Q$ *implies* $P^{'} \xrightarrow[\pi I]{\alpha} Q^{'}$ *where* $P = \varphi(P^{'})$ *and* $Q = \varphi(Q^{'})$ *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 it follows that* $P \equiv_\alpha P^{'}$ *and* $Q \equiv_\alpha Q^{'}$. *By applying the rule ALPHA in* πI, *we also obtain that* $P \xrightarrow[\pi I]{\alpha} Q$. *The procedure described in this Remark is trivial, and it is done only in our mind. We write just:* $P \xrightarrow[nI]{\alpha} Q$ *implies* $P \xrightarrow[\pi I]{\alpha} Q$.

5. Conclusion and Related Work

Notions as renaming, binding and fresh names appear in several approaches and it became evident that they deserve to be studied in their own terms. Nominal sets are related to the binding operators appearing in computer science, and represent an alternative set theory, with a more relaxed notion of finiteness. The theory of nominal sets was developed initially by Fraenkel and Mowstowski in 1930s. At that time, nominal sets were used to prove independence of the Axiom of Choice and other axioms in the classical ZF set theory. In computer science, they have been rediscovered by Gabbay and Pitts in [13] as an elegant formalism for modeling name binding. Since then, nominal sets have become a lively topic in semantics. Nominal sets were also independently rediscovered by the concurrency community, as a basis for syntax-free models of name-passing process algebras [18], and used in automata theory as a framework for describing automata on data words [9]. Computation in nominal sets have also been defined in [8]. In [6] the monoids defined in the category of nominal sets (also called nominal monoids) are used in the study of languages over infinite alphabets. The theory of syntactic monoids for languages of data words represents the same theory as the theory of finite monoids in the category of nominal sets, and under certain conditions, a language of data words is definable in first-order logic if and only if its syntactic monoid is aperiodic [7]. In [16] formal languages over infinite alphabets where words may contain binders are introduced. Nominal sets serve as a good framework for database theory since atoms can be used as an abstraction for data values, which can appear in a relational database or in an XML document. Atoms can also be used to model sources of infinite data in other applications, such as software verification, where an atom can represent a pointer or the contents of an array cell. Nominal algebraic structures are presented in terms of finitely supported objects in [1, 2]. The solution of the Scott recursive domain equation $D \cong (D \to D)$ in the FM framework is presented in [23].

This paper is an extended version of [4]. In this article we have rephrased the basic transition rules of the πI-calculus from [21] in a nominal form, and provide the nominal semantics of the πI-calculus. The transition rules in the nominal semantics of the πI-calculus are presented compactly, using a mixing of quantifiers ($\forall$, and the *nominal quantifier* $\mathsf{И}$) and not supplementary freshness conditions in their hypotheses. This

compact presentation of the transition rules is obtained thanks to the finite support property, Proposition 2.18, Proposition 2.19, and other technical results presented in the second section of the paper. Finally, we can prove that there is an equivalence between the nominal semantics of the πI-calculus and the usual (Sangiorgi) semantics of the πI-calculus presented in [21] (see Theorem 4.11).

There exists many papers where process calculi are modeled by using different techniques for binding. In [14], D. Hirschkoff formalised 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 [20] and in Coq [15]. 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 formalisations have to ensure that those terms are avoided. The nominal logic is a first order logic (see [19]) 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 [15]).

Nominal techniques have also been used in order to formalise the π-calculus. In [5] the π-calculus is formalised in Isabelle using the nominal datatype package [24]. The nominal techniques used in [5] are quite different from those proposed by us. The nominal quantifier is not used in [5] (the nominal quantifier can not be used to formalize the π-calculus in Isabelle). Our nominal semantics of the πI-calculus is based on the nominal quantifier which is used in order to "encode" the freshness conditions in the hypothesis of the transition rules of the πI-calculus. In paper [12], M.Gabbay also uses nominal techniques to obtain new transition rules for the π-calculus. Our goal is to provide nominal transition rules for the πI**-calculus**. We use the quantifier N to "encode" the freshness conditions in the hypothesis of the transition rules of the πI-calculus, and we obtain a set of compact transition rules which defines the nominal semantics of the πI-calculus. The main result of our paper states that the Sangiorgi semantics and the nominal semantics of the πI-calculus have the same expressive power. Our paper establishes an agreement between different presentations which, potentially, may break down for more powerful systems.

In [3] we used the nominal quantifier to express the transition rules of the π-calculus in the "weakest" form we need for describing the mobility. We made a refinement of the transition rule $L-CLOSE_L$ such that we assumed a similar rule to be valid only when some freshness conditions are satisfied. These freshness conditions are motivated by the manner in which the links are moved in a virtual space of linked processes. In the nominal semantics of the π-calculus defined in [3] (which is completely different from the related nominal semantics of the π-calculus defined in [12]), we can explain mobility using a transition rule which is assumed to be valid only in a more restrictive hypothesis than its related rule in the late semantics of the π-calculus. Even one transition rule in the nominal semantics of the π-calculus is assumed to be valid only in a more restricted hypothesis than its related rule in the

late semantics of the π-calculus, we are able to prove a complete equivalence between the nominal semantics and the late semantics of the π-calculus because of the finite support axiom.

References

[1] Alexandru A., Ciobanu G., *Nominal groups and their homomorphism theorems*, Fundamenta Informaticae, to appear.

[2] Alexandru A., Ciobanu G., *Nominal event structures*, Romanian Journal of Information, Science and Technology, **15**, pp. 79–90, 2012.

[3] Alexandru A., Ciobanu G., *Nominal semantics of mobility*, Romanian Journal of Information, Science and Technology, **15**, pp. 171–214, 2012.

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

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

[6] Bojanczyk M., *Data monoids*, Proc. 28th Symposium on Theoretical Aspects of Computer Science, pp. 105–116, 2011.

[7] Bojanczyk M., *Nominal monoids*, Theory of Computing Systems, **53**, pp. 194–222, 2013.

[8] Bojanczyk M., Braud L., Klin B., Lasota S., *Towards nominal computation*, Proc. 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, pp. 401–412, 2012.

[9] Bojanczyk M., Klin B., Lasota S., *Automata with group actions*, Proc. 26th IEEE Symposium on Logic in Computer Science, IEEE Computer Society Press, pp. 355–364, 2011.

[10] Gabbay M., *A study of substitution, using nominal techniques and Fraenkel-Mostowski sets*, Theoretical Computer Science, **410**, pp. 1159–1189, 2009.

[11] Gabbay M., *Foundations of nominal techniques: logic and semantics of variables in abstract syntax*, The Bulletin of Symbolic Logic, **17**, pp. 161–228, 2011.

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

[13] Gabbay M. J., Pitts A. M., *A new approach to abstract syntax with variable binding*, Formal Aspects of Computing, **13**, pp. 341–363, 2001.

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

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

[16] Kurz A., Suzuki T., Tuosto E., *On nominal regular languages with binders*, Proc. 15th International Conference on Foundations of Software Science and Computation Structures, LNCS 7213, pp. 255–269, 2012.

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

[18] Montanari U., Pistore M., *History-dependent automata: an introduction*, Proc. 5th International Conference on Formal Methods for the Design of Computer, Communication, and Software Systems: Mobile Computing, pp. 1–28, 2005.

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

[20] Rockl C., Hirschkoff D., *A fully adequate shallow embedding of the π-calculus in Isabelle/ HOL with mechanized syntax analysis*, Journal of Functional Programming, **13**, pp. 415–451, 2003.

[21] Sangiorgi D., *π-Calculus, Internal Mobility, and Agent-Passing Calculi*, Rapport INRIA no.2539, 1995.

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

[23] Shinwell M. R., *The Fresh Approach: Functional Programming with Names and Binders*, PhD Thesis, University of Cambridge, Computer Laboratory, February 2005.

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