Accepted for publication in Math. Struct. in Comp. Science

Mobile Petri Nets

A.Asperti and N.Busi

Received 25 June 2009

We add mobility to Place-Transition Petri Nets: tokens are names for places, and an input token of a transition can be used in its postset to specify a destination. Mobile Petri Nets are then further extended to Dynamic Nets, by adding the possibility of creating new nets during the firing of a transition. In this way, starting from Petri Nets, we define a simple hierarchy of nets with increasing degrees of dynamicity. For each class in this hierarchy we provide its encoding in the former class.
Our work has been largely inspired by the join-calculus of Fournet and Gonthier, that turns out to be a (well motivated) particular case of Dynamic Petri Nets. The main difference is that, in the preset of a transition, we allow for both non-linear patterns (name unification) and (locally) free names for input places (i.e. we remove the *locality* constraint, and preserve *reflexion*).

1. Introduction

Petri Nets are usually accepted as the main distributed model for concurrent computations. Unfortunately, Petri Nets are too static to be directly used as a specification language for distributed programming. In particular, they offer no direct way to express processes with changing structure, that is communicating agents that can be dynamically linked to other agents, possibly depending on previous communications. To bridge this gap, we define a hierarchy of nets with increasing degrees of dynamicity. The first step is to add mobility in the sense of (Milner, Parrow and Walker, 1992), namely the possibility to pass a reference to a process (a channel name) along a communication. From the point of view of Petri Nets, we can think of channels as places, and mobility amounts to considering tokens as names (actually, tuples of names) for places; then, the destinations in the postset of a transition can depend on the input tokens that have been read in the preset of the same transition. As a simple example, we consider a variant of the print-spooler in (Fournet and Gonthier, 1996). Available printers send their names and their type (color or black and white) to a place named `ready`, while users send their requests with the name of the file and the type of printer required to a place named `job`. For example, the configuration where the black and white printer named laser is ready, and we have two pending requests of printing the file `file1` on a black and white printer and the file `file2` on a color printer is described by the marking

$$ready(laser, bw), job(file1, bw), job(file2, c).$$

The print spooler matches a ready printer with a request and sends the file to the printer, as described by the following transition:

$$ready(PRINTER, TYPE), job(FILE, TYPE) \triangleright PRINTER(FILE).$$

Note the use of unification in the preset: the offer and the request match only if they have the same type. Firing this transition in the previous marking we would get the new configuration

$$job(file2, c), laser(file1).$$

The next step is to allow a transition to generate not only a new marking, but also a new set of transitions to be added to the system. This amounts to say that the postset of a transition is actually ... just another net!

In specifying this net we shall need a binding operator (νY) to distinguish the local names Y from non-local ones. When spawning a net during the firing of a transition, local names will be instantiated to fresh names, while non-local names will preserve their current meaning.

These nets will be called Dynamic Nets. As a simple example showing the expressive power of Dynamic Nets, let us consider the encoding of call-by-name λ-calculus.

Example 1. (call-by-name λ-calculus) A λ-term M is encoded as a net $(\nu v)[\![M]\!]_v$ where:

$$\begin{array}{rcl} [\![x]\!]_v &=& (\emptyset, x(v)) \\ [\![\lambda x.M]\!]_v &=& (\{v(x,u) \triangleright [\![M]\!]_u\}, \emptyset) \\ [\![(M\ N)]\!]_v &=& (\nu\{x,u\})([\![M]\!]_u \oplus (\{x(w) \triangleright [\![N]\!]_w\}, u(x,v))) \end{array}$$

The operator $\oplus$ denotes the sum of two nets defined in the obvious way (see Definition 5). Intuitively, you may think of v as the "root" of the term. A variable sends to the server for x its "root", that is the position where the actual value for x should be instantiated. A term $\lambda x.M$ waits "on its root" for two names x and u: the first name is the name of the server for the variable x, while u is the new "root" for M after the β-reduction. Finally, an application $(M\ N)$ creates two local names x and u. x is a server waiting for requests from variables: when a variable sends its "position" w to x, the server spawns a new instance of the argument N at position w. u is the root for M: on this channel, the application sends the name of the server and its own root (that, whenever M is "reduced" to $\lambda x.M'$, must become the new root for M'). The previous encoding is essentially borrowed from (Milner et al., 1992).

Dynamic Nets have been directly inspired by the join-calculus (Fournet and Gonthier, 1996), which in turns owes to the Chemical Abstract Machine (Berry and Boudol, 1992). The main difference between Dynamic Nets and this calculus is that, in the preset of a transition, we allow both non-linear patterns (name unification) and (locally) free names for input places. Using the terminology in (Fournet and Gonthier, 1996), we remove the locality constraints, and just keep reflexion. While locality has a clear implementative relevance (each subnet is an independent reaction-site which can be physically distributed without effort), the theoretical motivations behind this assumption are much less evident and should deserve a deeper investigation. Actually, the locality constraint of the join-calculus imposes a programming style that is not always intuitive and, in some cases, really frustrating. For example, the encoding of the λ-calculus in the join-calculus (regarded as a Dynamic Net) is the following one, which is considerably more contrived than the one given above:

$$\begin{array}{rcl}
[\![x]\!]_v & = & (\emptyset, x(v)) \\
[\![\lambda x.M]\!]_v & = & (\nu\{k\})(\{\{k(x,u) \triangleright [\![M]\!]_u\}, v(k)) \\
[\![(M\ N)]\!]_v & = & (\nu\{x,u\})([\![M]\!]_u \oplus (\{x(w) \triangleright [\![N]\!]_w, u(k) \triangleright k(x,v)\}, \emptyset))
\end{array}$$

2. Nets

In this section we recall the definition of Petri Nets, and give the formal definition of Mobile and Dynamic Nets.

2.1. *Petri Nets*

We recall the main definitions of Place/Transition nets without capacity constraints on places (see e.g. (Reisig, 1985)). We provide a characterization of this model using a notation that is convenient and consistent with our generalizations.

Definition 2. Given a set X, a multiset over X is a function $m : X \to (\mathcal{N} \cup \{\omega\})$. The set of all multisets over X is denoted by $\mathcal{M}_X$. Let $dom(m) = \{x \in X \mid m(x) > 0\}$. A multiset m is said to be empty if $m(x) = 0$ for all $x \in X$. Let

$$\mathcal{M}_X^{post} = \{m \in \mathcal{M}_X \mid dom(m) \text{ is finite}\}$$

and

$$\mathcal{M}_X^{pre} = \{m \in \mathcal{M}_X^{post} \mid m \text{ is not empty } \wedge \forall x \in X, m(x) \in \mathcal{N}\}.$$

Let $i < \omega$ and, for any $i \in \mathcal{N}$, $i+\omega = \omega+i = \omega+\omega = \omega$. We write $m \subseteq m'$ if $m(x) \leq m'(x)$ for all $x \in X$. The operator $\oplus$ denotes *multiset union*: $(m \oplus m')(x) = m(x) + m'(x)$. The operator $\setminus$ denotes *multiset difference*: $(m \setminus m')(x) = m(x) \ominus m'(x)$, where $\ominus$ is a partial operation over natural numbers defined as: $i \ominus j = i - j$ if $i > j$, $i \ominus j = 0$ if $i \leq j$, and $\omega \ominus i = \omega$.

Let X be a denumerable set of names, that will be used to indicate places in the net, ranged over by x, y, etc.

Definition 3. Let $N = (\nu Y)(T, m)$ where $Y \subseteq X$ is the set of places, $T \subseteq \mathcal{M}_X^{pre} \times \mathcal{M}_X^{pre}$ is the set of transitions, $m \in \mathcal{M}_X^{pre}$ is the initial marking. N is a P/T net if all names occurring in its initial marking and in its transitions are contained in the set of places, that is, if $dom(m) \cup \bigcup_{(c,p)\in T}(dom(c) \cup dom(p)) \subseteq Y$.

An element of $\mathcal{M}_X$ is called a marking. Given a marking m and a place x, we say that x *contains* $m(x)$ *tokens*. A transition $t = (c, p)$ will be written in the form $c \triangleright p$, where c is called the *preset* of t and represents the tokens to be "cconsumed"; p is called the *postset* of t and represents the tokens to be "produced".

Let $t = c \triangleright p$ be a transition: it is *enabled* at m if $c \subseteq m$; the execution of t enabled at m produces the marking $m' = (m \setminus c) \oplus p$. This is written as $m[t\rangle m'$.

2.2. *Mobile Nets*

Mobile nets are a variation of colored nets (Jensen, 1992), where the colors of the tokens are tuples of names. The new feature of mobile nets is the fact that the postset of the transitions is not static, but depends on the colors of the tokens the transition consumes. For instance, considering again the print-spooler example of the introduction, we can have a transition of the following kind:

$$ready(PRINTER, TYPE), job(FILE, TYPE) \triangleright PRINTER(FILE).$$

As in π-calculus and join-calculus we use names to represent both places and placeholders (variables) for names. In the example above, the upper case names are variables: they will be instantiated to actual names at the moment of firing. In general, given two names a and b, the notation $a(b)$ has a different meaning if it occurs in a marking or in the preset of a transition. In the former case b is an actual parameter, while in the latter, b is a formal parameter (that binds occurrences of the same name in the postset of the same transition). So the preset of a transition defines a pattern (possibly non linear) which has to be unified with a subset of the current marking to enable the transition.

Definition 4. Given two sets X and Y, let $\mathcal{M}_{X,Y} = X \to (Y \to (\mathcal{N} \cup \omega))$. Let $dom(m) = \{(x, y) \mid m(x)(y) > 0\}$,

$$\mathcal{M}^{post}_{X,Y} = \{m \in \mathcal{M}_{X,Y} \mid dom(m) \text{ is finite}\}$$

and

$$M^{pre}_{X,Y} = \{m \in \mathcal{M}^{post}_{X,Y} \mid m \text{ is not empty } \wedge \forall x \in X, y \in Y, m(x)(y) \in \mathcal{N}\}.$$

The operator $\oplus$ is defined as: $(m \oplus m')(x)(y) = m(x)(y) + m(x)(y)$. The operator $\setminus$ is defined as: $(m \setminus m')(x)(y) = m(x)(y) \ominus m(x)(y)$.

The set of colors of the tokens is defined as the set of finite (possibly empty) sequences on X : $\mathcal{C} = \{(x_1, \ldots, x_n) \mid n > 0 \wedge x_i \in X, i = 1, \ldots, n\}$.

In the following we use $\vec{x}, \vec{y}, \ldots$ to denote finite tuples of names. The *length* of a tuple is defined as $|(x_1, \ldots, x_n)| = n$ and the *selection* of the i-th element as $\pi_i(x_1, \ldots, x_n) = x_i$ for $i = 1, \ldots, n$. The operation of concatenation is represented by juxtaposition. With abuse of notation we use x instead of (x), when no confusion arises. Given a partial function ρ on X, we define substitution on names and on name tuples as:

$$x\rho = \begin{cases} y & \text{if } (x, y) \in \rho, \\ x & \text{otherwise,} \end{cases}$$

$$(x_1, \ldots, x_n)\rho = (x_1\rho, \ldots, x_n\rho).$$

Let $n(\rho) = \bigcup_{(x,y)\in\rho} \{x, y\}$.
Given $m \in \mathcal{M}_{X,\mathcal{C}}$, the substitution on all names in m is defined as

$$(m\rho)(x)(\vec{y}) = \sum_{v\rho = x \wedge \vec{z}\rho = \vec{y}} m(v)(\vec{z}),$$

whereas the substitution performed only on the names occurring in the colors of the tokens (used in the following for pattern instantiation), is

$$(m \star_b \rho)(x)(\vec{y}) = \sum_{\vec{z}\rho=\vec{y}} m(x)(\vec{z}).$$

Similarly, the definition of free and bound names is different in patterns (presets) and markings (postsets).
Free and bound names of m seen as a pattern are:

$$\begin{aligned}&fn_P(m) = \{x \mid \exists\vec{y}(m(x)(\vec{y}) > 0)\},\\ &bn_P(m) = \{x \mid \exists z, \vec{y}, i, \pi_i(\vec{y}) = x \wedge m(z)(\vec{y}) > 0).\}\end{aligned}$$

Free and bound names of m, seen as a marking, respectively as

$$\begin{aligned}&fn_M(m) = fn_P(m) \cup bn_P(m),\\ &bn_M(m) = \emptyset.\end{aligned}$$

In a transition, the bound names of the preset are binders for the postset. We define the *free names* of a transition t and of a set of transitions T respectively as

$$\begin{aligned}&fn(c \rhd p) = fn_P(c) \cup (fb_M(p) \backslash bn_P(c)),\\ &fn(T) = \textstyle\bigcup_{t\in T} fn(t).\end{aligned}$$

Definition 5. Let $N = (\nu Y)(T, m)$, where $Y \subseteq X$ is the set of places, $T \subseteq \mathcal{M}^{pre}_{X,\mathcal{C}} \times \mathcal{M}^{post}_{X,c}$ is the set of transitions and $m \in \mathcal{M}_{X,\mathcal{C}}$ is the initial marking. N is a *mobile net* if $fn_M(T) \cup fn(m) \subseteq Y$. An element of $M_{X,\mathcal{C}}$ is called a marking. Given a marking m, a place x and a color $\vec{y}$, we say that x *contains* $m(x)(\vec{y})$ tokens of color $\vec{y}$. A transition $t = (c, p)$ is usually written in the form $c \rhd p$. Let $t = c \rhd p$ be a transition: it is *enabled* at m if there exists $\rho \subseteq bn_P(c) \times X$ such that $c \star_b \rho \subseteq m$; the execution of t enabled at m with substitution ρ produces the marking $m' = (m \setminus c \star_b \rho) \oplus p\rho$. This is written as $m[t\rangle_\rho m'$.

The closure condition on the names in the net is a sufficient condition to guarantee that a name will not be used as a place name if it has not been declared in the set of places of the net.

We use the following concrete notation for markings: $m = x(\vec{y})$ is the marking with a single token of color $\vec{y}$ in the place x, that is $m(x)(\vec{y}) = 1$ and $m(v)(\vec{z}) = 0$ if $v \neq x$ or $\vec{y} \neq \vec{z}$; $m = \bigoplus^\infty x(\vec{y}) = x(\vec{y})^\infty$ is the marking with ω tokens of color $\vec{y}$ in the place x, that is $m(x)(\vec{y}) = \omega$ and $m(v)(\vec{z}) = 0$ if $v \neq x$ or $y \neq z$.

2.3. *Dynamic Nets*

A Dynamic Net is a Mobile Net where the set of places and transitions may increase during the execution: instead of producing only new tokens, a transition can generate a new subnet. As a consequence, the current state of the net is not represented any more by a marking, but by a net.

Definition 6. DN is the least set which satisfies the following equation:

$$\begin{aligned} X \quad = \quad & \{(\nu Y)(T,m) \mid \\ & Y \subseteq_{fin} X, \\ & T \subseteq_{fin} \{c \rhd N \mid c \in \mathcal{M}^{pre}_{X,\mathcal{C}}, N \in X\}, \\ & m \in \mathcal{M}_{X,\mathcal{C}}\}. \end{aligned}$$

Let $(\nu Y)(T,m) \in DN$; Y is the set of *places*, T the set of *transitions* and m the *marking.*

Besides the bound names in the preset of a transition, we have that also the names Y act as binders on (T,m) in $(\nu Y)(T,m)$. We define free names of transitions, sets of transitions and nets in the following way:

$$\begin{aligned} & fn(c \rhd N) = fn_P(c) \cup (fn(N) \setminus bn_P(c)), \\ & fn(T) = \cup_{t \in T} fn(t), \;\; fn((\nu Y)(T,m)) = (fn(T) \cup fn_M(m)) \setminus Y. \end{aligned}$$

In the definition of substitution on transitions and nets we must avoid that names intended to be free are captured by a binder; if the side condition is not satisfied we have to perform alpha conversion on the transition (or on the net) before.
Let $t = c \rhd N$ and $bn_P(c) \cap n(\rho) = \emptyset$; then

$$\begin{aligned} & t\rho = c\rho \rhd N\rho, \\ & T\rho = \{t\rho \mid t \in T\}. \end{aligned}$$

Let $N = (\nu Y)(T,m)$ and $Y \cap n(\rho) = \emptyset$; then

$$N\rho = (\nu Y)(T\rho; m\rho).$$

Let $N_1 = (\nu Y_1)(T_1, m_1)$ and $N_2 = (\nu Y_2)(T_2, m_2)$. If $Y_1 \cap Y_2 = \emptyset$, $fn(N_1) \cap Y_2 = \emptyset$; and $fn(N_2) \cap Y_1 = \emptyset$ we define

$$N_1 \oplus N_2 = (\nu Y_1 \cup Y_2)(T_1 \cup T_2; m_1 \oplus m_2).$$

If $Y_1 \cap Y_2 = \emptyset$ we define

$$(\nu Y_1)N_2 = (\nu Y_1 \cup Y_2)(T_2, m_2).$$

Definition 7. A *Dynamic Net* N is an element of the set DN that is closed, i.e. $fn(N) = \emptyset$. Let $N_1 = (\nu Y_1)(T_1, m_1)$ and $t = c \rhd N$ be a transition in T_1. We say that t is *enabled* at N_1 iff there exists $\rho \subseteq bn(c) \times X$ such that $c \star_b m_1 \subseteq m_1$; the execution of t enabled at N_1 with substitution ρ produces the new net $N_2 = (\nu Y_1)[(T_1, m_1 \setminus c \star_b \rho) \oplus N\rho]$. This is written as $N_1[t\rangle_\rho N_2$.

3. Encodings

In this section we define the encoding of Mobile Nets into Petri Nets, and of Dynamic Nets into Mobile Nets. The encoding is proved correct with respect to interleaving, but we claim that it also works for step and causal semantics.

3.1. *Encoding mobile nets into Petri nets*

We can simulate a mobile net with a Petri net in the following way: we represent the presence of a token with color $\vec{y}$ in the place x of the mobile net by means of a token in the place called $(x, \vec{y})$. Given a transition in the mobile net, for each possible instantiation of the bound names in its preset we provide a corresponding transition in the Petri net.

Let $N = (\nu Y)(T; m)$ be a mobile net. We will construct the corresponding Petri net N_{Petri}.

We first construct the set C of all tuples that may occur in the execution of N. Let $lenColors = \{|\vec{y}| \mid \exists x, m(x)(\vec{y}) > 0 \vee \exists (c \triangleright p) \in T, \exists x, c(x)(\vec{y} > 0 \vee p(x)(\vec{y}) > 0)\}$. If the set $lenColors$ has a maximum element n, then $C = \{\vec{x} \in C \mid |\vec{x}| \leq n\}$, else $C = \mathcal{C}$.

Now we define the mapping of a marking of N on a marking of N_{Petri}:

$$U(m)(x, \vec{y}) = m(x)(\vec{y}).$$

Given a transition t and an instantiation ρ of its bound names, the corresponding transition in N_{Petri} is

$$U(c \triangleright p, \rho) = U(c \star_b \rho) \triangleright U(p\rho).$$

Finally, let $N_{Petri} = (\nu Y_{Petri})(T_{Petri}, m_{Petri})$, where

$$\begin{aligned} &Y_{Petri} = Y \times C, \\ &T_{Petri} = \{U(t, \rho) \mid t \in T \wedge \rho \in X \to X\}, \\ &m_{Petri} = U(m). \end{aligned}$$

We have that N_{Petri} is a Petri net; moreover, each move in the mobile net N is matched by a move in the Petri net N_{Petri} and vice versa:

Theorem 8. Let $m_1, m_2 \in \mathcal{M}_{X,\mathcal{C}}$.
If $m_1[t\rangle_\rho m_2$ then $U(m_1)[U(t, \rho)\rangle U(m_2)$.
If $U(m_1)[t'\rangle m_2'$ then there exist t, ρ, m_2 such that $m_1[t\rangle_\rho m_2, t' = U(t, \rho)$ and $m_2' = U(m_2)$.

If the set of places and transitions of N are finite, then the set of places and transitions of N_{Petri} are finite.

3.2. *Encoding dynamic nets into mobile nets*

The translation of a dynamic net into a mobile net is a bit tricky. So, let us start with an example. Let us consider the following Net:

$$\begin{aligned} &(\nu\{A, B\})(\{A(X) \triangleright N'\}, A(A), A(B)), where \\ &N' = (\{X(W), Y(Z) \triangleright W(Z)\}, A(Y), Y(B)). \end{aligned}$$

The external net N has initial marking $A(A), A(B)$ and a single transition that reads a token X from A and spawns a new instance of the subnet N'. N' has initial marking $A(Y), Y(B)$ and contains a single transition

$$X(W), Y(Z) \triangleright W(Z).$$

The first (really rough) idea for transforming this net into a mobile net is to shift the internal transition to the external level (leaving the internal marking inside):

$$\begin{array}{ll}(\nu\{A,B,Y\}) & (\{A(X)\rhd A(Y),Y(B),\\ & \;X(W),Y(Z)\rhd W(Z)\},\\ & A(A),A(B)).\end{array}$$

Obviously, we have a lot of problems here, that we consider in turn. First of all, the internal transition is now always (potentially) enabled, while it should be activated by the firing of the external one. Thus, we need an explicit enabling place for each subnet; we also use this place to pass actual parameters for the free names in the subnet-transitions. For the sake of uniformity, we also add an enabling place for the initial net. According to this idea, our pseudo-mobile net is now modified as follows:

$$\begin{array}{ll}(\nu\{A,B,Y\}) & (\{en_N(),A(X)\rhd A(Y),Y(B),en_{N'}(X)^{\infty}\\ & \;en_{N'}(X),X(W),Y(Z)\rhd W(Z)\},\\ & A(A),A(B),en_N()^{\infty}).\end{array}$$

The second problem is that the spawning process should generate new instances at each firing of the external rule. So, we must use new names to distinguish between these different instances. In particular, a name X of a channel should become a pair $X, X\delta$ where $X\delta$ denotes the particular instance of the channel X in use. These names are taken from a tank (infinite supplier for fresh names) at the moment of the firing.
When we send an information to a channel A, we shall also pass as first component of the message the particular instance of A we are referring to.

$$\begin{array}{ll}(\nu\{A,B,Y\}) & (\{en_N(this),A(this,X,X\delta),tank(new)\rhd\\ & \qquad A(this,Y,new),Y(new,B,this),en_{N'}(new,X,X\delta)^{\infty},\\ & \;en_{N'}(this,X,X\delta),X(X\delta,W,W\delta),Y(Y\delta,Z,Z\delta)\rhd W(W\delta,Z,Z\delta)\},\\ & A(v_0,A,v_0),A(v_0,B,v_0),en_N(v_0)^{\infty},\bigoplus_{i>0}tank(v_i)).\end{array}$$

The last problem is that the second transition is not yet a mobile transition in our sense. The problem is the receiving channel on X. The obvious idea would be to consider all its possible instantiations, but in this way we would activate too many transitions, instead of the single one whose name is received from the enabler. The solution is to slightly modify the structure of the tokens, in such a way that they testify the place they belong to: so each token in a place A will always be a tuple starting with A. Now, we can instantiate X to an arbitrary (known) place of the net, still being sure that it will actually consume tokens only from the place whose name has been indicated by the enabler. Let us do it in two steps. First we enrich the structure of tokens:

$$\begin{array}{ll}(\nu\{A,B,Y\}) & (\{en_N(this),A(A,this,X,X\delta),tank(new)\rhd\\ & \qquad A(A,this,Y,new),Y(Y,new,B,this),en_{N'}(new,X,X\delta)^{\infty},\\ & \;en_{N'}(this,X,X\delta),X(X,X\delta,W,W\delta),Y(Y,Y\delta,Z,Z\delta)\rhd\\ & \qquad W(W,W\delta,Z,Z\delta)\},\\ & A(A,v_0,A,v_0),A(A,v_0,B,v_0),en_N(v_0)^{\infty},\bigoplus_{i>0}tank(v_i)).\end{array}$$

Next, we consider all possible instances of the second transition by substituting X for one of the places A, B or Y:

$$
\begin{aligned}
(\nu\{A,B,Y\}) \quad (\{& en_N(this), A(A, this, X, X\delta), tank(new) \triangleright \\
&\quad A(A, this, Y, new), Y(Y, new, B, this), en_{N'}(new, X, X\delta)^\infty, \\
& en_{N'}(this, X, X\delta), A(X, X\delta, W, W\delta), Y(Y, Y\delta, Z, Z\delta) \triangleright \\
&\quad W(W, W\delta, Z, Z\delta)\}, \\
& en_{N'}(this, X, X\delta), B(X, X\delta, W, W\delta), Y(Y, Y\delta, Z, Z\delta) \triangleright \\
&\quad W(W, W\delta, Z, Z\delta)\}, \\
& en_{N'}(this, X, X\delta), Y(X, X\delta, W, W\delta), Y(Y, Y\delta, Z, Z\delta) \triangleright \\
&\quad W(W, W\delta, Z, Z\delta)\}, \\
& A(A, v_0, A, v_0), A(A, v_0, B, v_0), en_N(v_0)^\infty, \textstyle\bigoplus_{i>0} tank(v_i)).
\end{aligned}
$$

Let us provide an example of token game in the two nets. The dynamic net has initial marking $A(A), A(B)$. By firing the (unique) transition with input token $A(A)$ we get the new marking $A(B), A(Y_1), Y_1(B)$, where Y_1 is a fresh name. Moreover, the transition

$$A(W), Y_1(Z) \triangleright W(Z)$$

is now added to the system. We have several possibilities, now. Suppose to fire again the "external" transition with input token $A(Y_1)$. The marking becomes $A(B), Y_1(B), A(Y_2), Y_2(B)$ and the new transition

$$Y_1(W), Y_2(Z) \triangleright W(Z)$$

is activated. Now we can fire this transition, getting the marking $A(B), A(Y_2), B(B)$. The same firing sequence is simulated in the mobile net by the following steps:

$$
\begin{gathered}
A(A, v_0, A, v_0), A(A, v_0, B, v_0), en_N(v_0)^\infty, \textstyle\bigoplus_{i>0} tank(v_i) \\
\Rightarrow \\
A(A, v_0, B, v_0), A(A, v_0, Y, v_1), Y(Y, v_1, B, v_0), \\
en_N(v_0)^\infty, en_{N'}(v_1, A, v_0)^\infty, \textstyle\bigoplus_{i>1} tank(v_i) \\
\Rightarrow \\
A(A, v_0, B, v_0), Y(Y, v_1, B, v_0), A(A, v_0, Y, v_2), Y(Y, v_2, B, v_0), \\
en_N(v_0)^\infty, en_{N'}(v_1, A, v_0)^\infty, en_{N'}(v_2, Y, v_1)^\infty \textstyle\bigoplus_{i>2} tank(v_i) \\
\Rightarrow \\
A(A, v_0, B, v_0), A(A, v_0, Y, v_2), B(B, v_0, B, v_0), \\
en_N(v_0)^\infty, en_{N'}(v_1, A, v_0)^\infty, en_{N'}(v_2, Y, v_1)^\infty \textstyle\bigoplus_{i>2} tank(v_i)
\end{gathered}
$$

3.3. *The formal definition*

Given a net $N = (\nu Y)(T, m)$, we denote Y with $locals_N$, T with $trans_N$ and m with $mark_N$.

Definition 9. Let $N_1, N_2 \in DN$. N_1 occurs in N_2 iff $N_1 = N_2$ or there exist c, N such that $(c \triangleright N) \in T_2$ and N_1 occurs in N. $N1$ occurs properly in N_2 iff N_1 occurs in N_2 and $N_1 \neq N_2$.

Let $N = (\nu Y)(T, m)$ be a dynamic net. We assume that the names occurring in the binders are all different. This condition can be easily fulfilled by performing an alpha

conversion that substitutes each bound name with a fresh name.
We now construct a corresponding mobile net N_M.
For each $N' = (\nu Y')(T', m')$ occurring in N, let

— $en_{N'}$ be a fresh name
— $free_{N'}$ be a sequence containing exactly one occurrence of every name in $fn(T') \setminus Y'$

$en_{N'}$ is a place used to enable the corresponding net N'; the color of tokens in $en_{N'}$ will represent the current instantiation of the free names of N' occurring in the transitions of N'.

Let $Places = \cup_{N' \text{ occurs in } N} locals_{N'}$; this will be the subset of places in N_M that correspond to places in N.

Let $tank; v_i$ for $i \in \mathcal{N}$ be fresh names; the tank will contain a token for each color v_i, for $i > 0$, whereas the name v_0 will be associated with the names occurring in the (unique) instance of the net N.

Given a function $\delta X \to X$ that associates to each name a name-instance, we proceed to map markings in N to markings in N_M. Given, a marking $m \in \mathcal{M}_{X,\mathcal{C}}$, let us define:

$$\begin{aligned}
&d^\delta(x_1, \ldots, x_n) = x_1, x_1\delta, \ldots, x_n, x_n\delta,\\
&D^\delta(m)(x)(x, x\delta, y_1, y_1\delta, \ldots, y_n, y_n\delta) = m(x)(y_1, \ldots, y_n),\\
&D^\delta(m)(x)(y_0, y_0\delta, y_1, y_1', \ldots, y_n, y_n') = 0 \text{ if } y_0 \neq x \vee \exists i, 1 \leq i \leq n, y_i' \neq y_i\delta.
\end{aligned}$$

Let X' be a denumerable set of fresh names. Let $\delta_0 : X \to X'$ be a bijection.
Now we transform each transition occurring in N in a set of transitions in N_M.
Let $this, new$ be fresh names, and $t = c \triangleright N'' \in trans_{N'}$. Let

$$\begin{aligned}
\delta(x) &= \begin{cases} this & \text{if } x \in loc_{N'} \\ new & \text{if } x \in loc_{N''} \\ \delta_0(x) & \text{otherwise} \end{cases}\\
D_1(t) &= en_{N'}(this, d^\delta(free_{N'}) \oplus D^\delta(c) \oplus tank(new) \triangleright\\
&\quad D^\delta(m'') \oplus \bigoplus^\infty en_{N''}(new, d^\delta(free_{N''})).
\end{aligned}$$

Let $(c \star_f \rho) = \sum_{\rho(z)=x} c(x)(\vec{y})$ (substitution on place names only);

$$\begin{aligned}
D_2(c \triangleright p, \rho) &= c \star_f \rho \triangleright p,\\
D_3(t, \rho) &= D_2(D_1(t), \rho),\\
D_4(N') &= \{D_3(t, \rho) \mid t \in trans_{N'} \wedge\\
&\qquad \rho \subseteq (fn_P(c) \cap free_{N'}) \times Places.\}
\end{aligned}$$

Let

$$\delta_1(x) = \begin{cases} v_0 & \text{if } x \in locals_N, \\ \delta_0(x) & \text{otherwise.} \end{cases}$$

The mobile net corresponding to N is $N_M = (\nu Y_M)(T_M, m_M)$, where

$$\begin{aligned}
&Y_M = Places \cup \{tank\} \cup \{en_{N'} \mid N' occurs in N\},\\
&T_M = \textstyle\sum_{N' \text{ occurs in } N} D_4(N'),\\
&m_M = D^{\delta_1}(m) \oplus \bigoplus_{i>0} tank(v_i) \oplus \bigoplus^\infty en_N(v_0).
\end{aligned}$$

In our encoding we need transitions that put infinitely many tokens of the same color in a place: to simulate the generation of a new subnet from the firing of a transition in N, the corresponding transition in N_M puts an infinite quantity of tokens in the enabling place corresponding to the subnet. We can avoid such a kind of transitions if we extend our model to *contextual nets* (Montanari and Rossi, 1995), where a transition can "read" a token from a place without consuming it.

For uniformity reasons, in our encoding we add an enabling place also for the initial net and for subnets with an empty set of transitions; we can avoid to add these redundant places, thus obtaining a more compact net.

Let $N = (\nu Y)(T, m)$. To relate the behaviour of N with the behaviour of N_M we need to decorate the nets properly occurring in N with some information linking them to the corresponding part of N_M. To this aim, we decorate each of these nets with the corresponding enabling place and two occurrences of the names free in its transitions; one of these occurrences will be modified by the instantiations performed on the net, whereas the other is left unchanged. From these information we obtain the actual instantiation of the free names when a copy of the net is generated. Moreover, we decorate each transition of these nets with all the corresponding transitions in N_M (obtained by instantiation of the free names), recording for each of them the instantiation it originates from. The last information is used to know which one of the corresponding transitions will be enabled at the generation of the net.

The decorated form of N is $(\nu Y)(\{c \triangleright dec(N) \mid c \triangleright N \in T\}, m)$, where:

$$\begin{aligned}
&dec((\nu Y)(T, m)) = ((\nu Y)(dec(T, N), m); free_N; en_N; free_N),\\
&dec(T, N) = \cup_{t \in T} dec(t, N),\\
&dec(c \triangleright N', N) = \{(c \triangleright dec(N'), D_3(c \triangleright N', \rho), \rho)) \mid \rho \in (fn_P(c) \cap free_N) \rightarrow Places\}.
\end{aligned}$$

The substitution on decorated nets is defined in this way:

$$\begin{aligned}
&(N, \vec{x}, en, \vec{y})\rho = (N\rho, \vec{x}\rho, en, \vec{y}),\\
&(t, t', \sigma) = (t\rho, t', \sigma).
\end{aligned}$$

Given a transition $t = c \triangleright (N'', \vec{x}; en; \vec{y})$, the firing rule on decorated nets is: $N[t\rangle_\rho N'$ iff $c \star_b \rho \subseteq m$ and $N' = (\nu Y)((T, m \setminus c \star_b \rho) \oplus N'''\rho)$, where $N''' = (\nu Y'')(\pi_1(T''), m'')$.

The relation between the current state of the net N and the net N_M is recorded by a tuple $R = (V, En, P, Tr)$, where

— V is a set of names corresponding to the current content of the tank;

— En contains elements of the form $(en, v, \vec{z})$, meaning that N_1 contains a subnet whose corresponding enabling place and instance in N_M are respectively en and v, and when it has been generated its free names were instantiated to $\vec{z}$;

— P contains elements of the form $(x, (y, v))$, meaning that the place x in N_1 corresponds with the instance v of the place y in the net N_M;

— Tr contains elements of the form $(t, (t', v, \rho)))$, associating the transition t of N_1 with the instance v of the transition t in N_M, and ρ gives the instantiation of each free name with its actual value when t has been generated.

The initial relation between the net N and the initial marking of N_M is $R_0 = (V_0, En_0, P_0, Tr_0)$, where

$$\begin{array}{l} V_0 = \{v_i \mid i > 0\}, \\ En_0 = \{(en_N, v_0, ())\}, \\ P_0 = \{(y, (y, v_0)) \mid y \in loc_N\}, \\ Tr_0 = \{(t, (D_1(t), v_0, \emptyset)) \mid t \in trans_N.\} \end{array}$$

Given a current state N_1 of the net N and a corresponding relation $R = (V, En, P, Tr)$, we map the marking $m = mark_{N_1}$ in a marking $F_R(m)$ of the net N_M in the following way:

$$\begin{array}{l} F_R(m)(tank)(v) = \begin{cases} 1 & \text{if } v \in V, \\ 0 & \text{otherwise.} \end{cases} \\ F_R(m)(en)(v, (Pz_1)\dots(Pz_n)) = \begin{cases} 1 & \text{if } (en, v, z_1, \dots, z_n) \in En, \\ 0 & \text{otherwise.} \end{cases} \\ F_R(m)(\pi_1(Px))((Px)(Py_1)\dots(Py_n)) = m(x)(y_1, \dots, y_n). \\ F_R(m)(x)(y_1, y_1', \dots, y_n, y_n') = 0 \text{ if } x \neq y_1 \vee \exists i, 1 \le i \le n \wedge \forall x, (x, (y_i, y_i')) \notin P. \end{array}$$

We map a substitution ρ referring to tokens in N_1 to the corresponding substitution $S_R(\rho)$ on tokens in N_M in this way:

$$S_R(\rho) = \cup_{(x,y)\in\rho}\{(x, \pi_1(Py)), (\delta_0(x), \pi_2(Py))\}.$$

Now we are ready to state the correspondence between N and N_M.

Theorem 10. Let N_1 be the current state of the net N and $R = (V; En; P; Tr)$ the associated relation.

— if $N_1[t\rangle_\rho N_2$, with $t = c \triangleright (N_3, \vec{x}, en, \vec{y})$, then there exists $(t, t_M, v_{this}, \tau) \in Tr$ such that $F_R(mark_{N_1})[t_M\rangle\rho_M F_{R'}(mark_{N_2})$, where, given $v_{new} \in V$,

$$\rho_M = \{(this; v_{this}, (new, v_{new})\} \cup S_R(\rho \cup \tau)$$

and $R = (V', En', P', Tr')$, with

$$\begin{array}{l} V' = V \setminus \{v_{new}\}, \\ En' = En \cup \{(en; v_{new}; \vec{x})\}, \\ P' = P \cup \{(\sigma(y), (y, v)) \mid y \in local_{N_3}\}, \\ Tr' = Tr \cup \{(t, (t', v, \{\vec{x}/\vec{y}\}) \mid \exists\rho, (t, t', \rho) \in trans_{N_4} \wedge \rho \subseteq \{\vec{x}/\vec{y}\})\}, \end{array}$$

where N_4 is the net obtained by alpha conversion of the place names in N_3 with the substitution σ, that has been added to N_1 to obtain N_2.

— If $F_R(mark_{N_1})[t_M\rangle_{\rho_M} m_M'$ then there exists $(t, t_M, \rho_M(this), \tau) \in Tr$, with $t = c \triangleright (N_3, \vec{x}, en, \vec{y})$, such that $N_1[t\rangle_\rho N_2$ and $m_M' = F_{R'}(mark_{N_2})$, where

$$\rho = \{(x, y) \mid x \in bn_P(c) \wedge \exists z_1, z_2, (x, z_1) \in \rho_M \wedge (y, (z_1, z_2)) \in P\}$$

and $R' = (V', En', P', Tr')$, with

$$\begin{aligned}
&V' = V \setminus \{\rho_M(new)\},\\
&En' = En \cup \{(en, \rho_M(new), \vec{x})\},\\
&P' = P \cup \{(\sigma(y), (y, v)) \mid y \in local_{N_3}\},\\
&Tr' = Tr \cup \{(t, (t', v, \{\vec{x}/\vec{y}\}) \mid \exists \rho, (t, t', \rho) \in trans_{N_4} \wedge \rho \subseteq \{\vec{x}/\vec{y}\})\},
\end{aligned}$$

where N_4 is the net obtained by alpha conversion of the place names in N3 with the substitution σ, that has been added to N_1 to obtain N_2.

4. Conclusions

We have enriched Petri Nets with mobility (Mobile Nets) and reflexion (Dynamic Nets). We both propose Dynamic Nets as a new foundational model of concurrency and a formal basis for a specification language for distributed programming. From the theoretical point of view a lot of work is obviously left. We claim that our encoding of Dynamic Nets into Petri Nets should also work for step and process semantics, but these notions should deserve a better investigation before proving the theorem. In the spirit of Dynamic Nets as a specification language, it would also be interesting to study observational semantics. We are currently investigating a higher-order extension of our nets, that is the possibility to have nets as token colors, that would allow explicit transmission of processes (Sangiorgi, 1993). We believe that we could use techniques similar to those described in this paper to translate these Higher-order Nets into Dynamic Nets. Another interesting extension is to permit recursive definitions, which are not covered by the current approach. Finally, we would like to acknowledge again the great influence of the join-calculus in the definition of Dynamic Nets. The proper subset of Dynamic Nets corresponding to terms of the join-calculus seems to have nice properties of locality that would be interesting to study in more detail (for instance their encoding into Mobile Nets is much simpler than for the general case). However, Dynamic Nets seem to provide a higher degree of dynamicity than the join-calculus; hence the encoding of Dynamic Nets into the join-calculus would provide an interesting test-case to assess the actual expressivity of the latter formalism. Even if a simple encoding could be written, it could still be better to use Dynamic Nets at the specification level, considering the join calculus as a sort of intermediate “machine” language towards a really distributed implementation.

References

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

Fournet, C. and Gonthier, G. (1996). The reflexive cham and the join-calculus, *POPL*, pp. 372–385.

Jensen, K. (1992). *Coloured Petri Nets*, EATCS Monographs in Computer Science, Springer Verlag.

Milner, R., Parrow, J. and Walker, D. (1992). A calculus of mobile processes, *Inf. Comput.* **100**(1): 1–77.

Montanari, U. and Rossi, F. (1995). Contextual nets, *Acta Inf.* **32**(6): 545–596.

Reisig, W. (1985). *Petri Nets: An Introduction*, EATCS Monographs in Computer Science, Springer Verlag.

Sangiorgi, D. (1993). *Expressing Mobility in Process Algebra*, Ph.d. thesis, University of Edinburgh.