

A Combined *Open Petri Net* and *Process Algebraic* approach to Message-Passing Services*

Chrysafis Hartonas

Dept of Computer Science, TEI Larissa,
41110 Larissa, Greece
hartonas@cs.teilar.gr

Abstract. Two of the significant issues in *Service Oriented Architecture* (SOA) are, first, the compositional analysis (and construction) of services and, second, the analysis and modeling of service communication. Existing formalisms to address these issues that have been investigated include petri nets and process algebras. The objective of this report is to delineate an appropriate class of petri nets, suitable for the modeling of communicating processes (services) with value (message) passing, extending our results in [3,4]. This involves defining appropriate operators, such as prefix, choice and, more significantly, concurrent composition. It also involves demonstrating how recursive equations can be solved in such a class of petri nets, providing the necessary semantic background for recursively definable processes.

Keywords Open Petri Nets, Value Passing, Communicating Processes, Service Oriented Architecture.

1 Preliminaries

Two of the significant issues in *Service Oriented Architecture* (SOA) are, first, the compositional analysis (and construction) of services and, second, the analysis and modeling of service communication. Existing formalisms to address these issues that have been investigated include petri nets and process algebras. Whereas petri net models provide a natural and intuitive account of concurrent composition (that can be used to model service communication, see e.g. [4]), petri nets are not designed to naturally support a compositional analysis of processes. By contrast, process algebras are compositional in approach, though less intuitive than petri nets. Some work on combining the strengths of the two

* Research conducted in the context of the PerLA and the AgentNet projects, led by this author. PerLA is co-funded by the European Social Fund and National Resources (EPEAEK II) ARCHIMEDES II. AgentNet is part of a larger research project run by the Graduate Program in Logic, Algorithmics and the Theory of Computation of the University of Athens and co-funded by the European Social Fund and National Resources (EPEAEK II) PYTHAGORAS II.

approaches has been done, e.g. [3, 9] and the present reports falls within the line of research that addresses issues of process algebraic and petri-net based approaches to value-passing, communicating processes (or services).

The objective of this report is to delineate an appropriate class of petri nets, suitable for the modeling of communicating processes (services) with value (message) passing. This involves defining appropriate operators on such a class of petri nets, such as prefix, choice and, more significantly, concurrent composition. It also involves demonstrating how recursive equations can be solved in such a class of petri nets, providing the necessary semantic background for recursively definable processes.

2 Communicating Petri Nets

For reader's convenience, we first briefly recall the definition of place-transition nets.

Definition 1 (Place-Transition Petri Nets). Assume disjoint and nonempty sets $Place$ and $Trans$ of place and transition labels, respectively. A *Petri Net* with place labels in $P \subseteq Place$ and transition labels in $T \subseteq Trans$ is a triple

$$\mathcal{A} = (P, \longrightarrow, M_0)$$

where $M_0 \subseteq \mathcal{P}_{nf}(P)$ is called the *initial marking*, and $\longrightarrow \subseteq \mathcal{P}_{nf}(P) \times T \times \mathcal{P}_{nf}(P)$ is the transition relation (the subscript *nf* indicates non-empty finite subsets). A transition $t = (I, a, O)$ will be also denoted by $t = I \xrightarrow{a} O$. The set $I = \mathbf{pre}(t)$ is called the preset of the transition, while $O = \mathbf{post}(t)$ its postset, and $\mathbf{act}(t) = a$ its action. We occasionally also write $Pl(\mathcal{A})$ for the set P of places of \mathcal{A} . \square

Note that, for simplicity, we restrict here to markings with single tokens on places, but this is an inessential restriction.

2.1 Open Petri Nets and Net Firings

To define open nets, assume pairwise disjoint sets $Place$, $Trans$ and $Port$.

Definition 2 (Open Petri Nets). An *open net* is like an ordinary net except for

1. its set of places is contained in the union $Place \cup Port$
2. its action labels are either
 - τ (a silent, internal action), or
 - $c!$ or $c?$, with $c \in C$, a set of communicative action names
3. if $I \xrightarrow{c!} O$ is a transition in the net, then $I \cap Port = \emptyset$ and $O = O' \cup \{x\}$, with $\emptyset \neq O' \subseteq Place$ and $x \in Port$
4. if $I \xrightarrow{c?} O$ is a transition, then $I \cup O \subseteq Place$ and, for some port name $x \in Port$ and set of places $U \subseteq Place$, there is a transition $O \cup \{x\} \xrightarrow{\tau} U$

5. if $I \xrightarrow{\tau} O$ is a transition, then $O \subseteq \text{Place}$ and, unless there is a transition $J \xrightarrow{c?} K$ such that $K \cap I \neq \emptyset$, the preset I of τ contains no ports (i.e. $I \cap \text{Port} = \emptyset$) \square

Intuitively, a transition $t = I \xrightarrow{c!} O \cup \{x\}$ is a transition that executes an asynchronous output action and deposits the output on a communication buffer, named x . The action is asynchronous in the sense that the net will proceed to fire any transitions that became enabled after firing t , without waiting for its output to be consumed. Similarly, a transition $r = J \xrightarrow{c?} K$ is an action that prompts (requests) input on some specific channel c . If such input is provided, through some buffer y associated to channel c , then it is consumed by an internal action $\{y\} \cup K \xrightarrow{\tau} L$.

Note that we have restricted only to visible, communicating actions and any internal action (process) of the service is uniformly designated by a τ action.

It should be also noted that we focus on the communication exchange per se, ignoring the type of content being communicated. Thus, in this setting, what is passed from service to service during communication is merely a “token”.

In the sequel, we write $\mathcal{A} \cong \mathcal{B}$ for two nets, provided there is an injective renaming σ of place (including port) names, such that $\mathcal{A}\sigma = \mathcal{B}$ (where $\mathcal{A}\sigma$ is the net obtained by applying the renaming σ to the places of \mathcal{A}).

In the next Section we recall the standard notion of “enabled transitions” and net firings and, furthermore, we extend to appropriate notions of *weakly enabled transitions* and *weak net firing*. The intention here is to capture the (potential) behavior of an open net, assuming that any requested input becomes available when needed.

Definition 3. A transition t is enabled at a marking M if $\text{pre}(t) \subseteq M$ (i.e. all places and ports in its preset hold a token). It is weakly enabled if all places (but not necessarily the ports) in $\text{pre}(t)$ are contained in the marking, in other words, if $\text{pre}(t) \subseteq M \cup (\text{pre}(t) \cap \text{Port})$.

Two transitions t, t' are concurrently enabled, if they are both enabled and $\text{pre}(t) \cap \text{pre}(t') = \emptyset$. They are weakly, concurrently enabled if they are both weakly enabled and $\text{pre}(t) \cap \text{pre}(t') = \emptyset$. \square

The definition of (weakly) concurrently enabled transitions generalizes in the obvious way to more than two transitions.

A (one-step) *firing* of a net is a transformation of its marking, such that, if T is a set of concurrently enabled transitions, then a non-empty subset S of T is (non-deterministically chosen and) “fires”. Intuitively, “firing” of a transition results in removing the tokens from its **pre** set and depositing them in its **post** set. Technically, if $t = I \xrightarrow{a} O$ is enabled at the marking M , then firing the transition results in the new marking $M' = (M \setminus I) \cup O$ and we write $M \xrightarrow{a} M'$. Similarly for a set $S = \{t_1, \dots, t_n\}$ of concurrently enabled transitions we write $\text{act}(S) = \text{act}(t_1) \cdots \text{act}(t_n)$ and $M \xrightarrow{\text{act}(S)} M'$, where $M' = (M \setminus \bigcup_i \text{pre}(t_i)) \cup \bigcup_i \text{post}(t_i)$.

A sequence of firings

$$M = M_0 \xrightarrow{\text{act}(T_1)} M_1 \xrightarrow{\text{act}(T_2)} \dots \xrightarrow{\text{act}(T_n)} M_n \xrightarrow{\text{act}(T_{n+1})} \dots$$

will be called a *firing sequence*.

A *weak firing sequence* is defined similarly, assuming the T_i 's are weakly enabled sets of transitions and making the input assumptions explicit by listing the input port as if marked.

Definition 4. A place A of the net is (weakly) reachable, given the marking M , if there exists a finite sequence of firings of (weakly) concurrently enabled sets of transitions $T_i = \{t_{i,1}, \dots, t_{i,m(i)}\}$

$$M = M_0 \xrightarrow{\text{act}(T_1)} M_1 \xrightarrow{\text{act}(T_2)} \dots \xrightarrow{\text{act}(T_n)} M_n$$

such that $A \in M_n$ (for some $n \geq 0$). □

The reachability of the states (weak, or otherwise) can be represented with a *reachability graph*, a labeled, directed graph (a *transition system*) whose points represent states (sets of marked places, i.e. markings), and arcs represent transitions between two such states.

Before proceeding to define the *concurrent join* of nets (including rigorous definitions for the notions of *rank numbers* and *rank sequence*, we mention a simple notational convention. For a set T of concurrently (weakly) enabled transitions, we write $\text{act}^-(T)$ for the sequence of communication actions in $\text{act}(T)$. If no such actions exist in $\text{act}(T)$, we let $\text{act}^-(T) = 1$. Thus, if $\text{act}(T) = \tau c! \tau \tau d?$, then $\text{act}^-(T) = c! d?$ and if $\text{act}(T) = \tau^n$, then $\text{act}^-(T) = 1$.

2.2 Composable Open Petri Nets

The main issue in composing open nets is to formally identify *composable transitions* in the two nets. We have done this in detail in [4]. For reader's convenience we briefly review from [4] the relevant definitions of *rank sequence* and *matching transitions*.

Intuitively, the rank sequence (t) of a communication transition t with action $c!$ (similarly for $c?$) consists of rank numbers of order $k \geq 1$, denoted by $(t)_k$, where k indicates the number of times t became enabled in a weak firing sequence for which t becomes enabled at the earliest possible time. More specifically, if $\text{act}(t) = c!$ (and similarly for $c?$), $(t)_k = i$ means that when enabled for the k -th time, exactly $i - 1$ transitions with action $c!$ have become enabled to fire, up to the point where the transition t becomes enabled for the k -th time, at the earliest possible opportunity. The formal definition follows.

Definition 5 (Rank Sequence). The rank sequence of a communication transition t , denoted by (t) , is the increasing sequence of its rank numbers.

A natural number $i \geq k$ is a σ -rank-number of order $k \geq 1$ for t , if σ is a weak firing sequence

$$M = M_0 \xrightarrow{\text{act}^-(T_1)} M_1 \xrightarrow{\text{act}^-(T_2)} \dots \xrightarrow{\text{act}^-(T_n)} M_n \xrightarrow{\text{act}^-(T_{n+1})}$$

such that

- there exist $j_1, \dots, j_k \in \{1, 2, \dots\}$ such that
 - if $a < b$, then $j_a < j_b$
 - t is weakly enabled (concurrently with other transitions) in each of T_{j_1}, \dots, T_{j_k}
 - if $s < j_b$, for some b , and t is weakly enabled (concurrently with other transitions) in T_s , then $s = j_a$, for some $a < b$
- if $\mathbf{act}(t) = c!$ (and similarly if $\mathbf{act}(t) = c?$), then there are exactly $i - 1$ transitions $t^{(1)}, \dots, t^{(i-1)}$ with $\mathbf{act}(t^{(r)}) = c!$, for each $r = 1, \dots, i - 1$ such that $t^{(r)} \in T_{j_i}$ for some j_i .

The number i is the rank number of order k for t (denoted by $i = (t)_k$), provided that i is the least σ -rank-number of order k for t . \square

For details and examples the reader is referred to our [4].

Now let \mathcal{A}, \mathcal{B} be two nets and $t = I \xrightarrow{c!} O \cup \{x\}$ an output transition in \mathcal{A} . Consider its rank sequence (t) . The rank number $(t)_1$ designates the number of output transitions that have become enabled up to, including, the point when t becomes enabled (at the earliest possible opportunity) for the first time. Turning to \mathcal{B} , let $r = J \xrightarrow{c?} K, K \cup \{y\} \xrightarrow{\tau} L$ be an input prompt transition, followed by a transition that consumes the input when provided. Consider its rank sequence (r) . Assuming the nets are to be composed concurrently, under what conditions should it be possible and permissible for output from t to be fed as input to r ? The Definition that follows captures our proposed notion of *matching transitions* and, therefore, of *composable services*.

Definition 6 (Matching Transitions). Let \mathcal{A}, \mathcal{B} be two nets, $t = I \xrightarrow{c!}_A O \cup \{x\}$ an output transition in \mathcal{A} and $r = J \xrightarrow{c?}_B K, K \cup \{y\} \xrightarrow{\tau}_B L$ be an input prompt transition in \mathcal{B} , followed by a transition that consumes the input when provided. If there exist $m, k \leq \min\{\mathbf{length}(t), \mathbf{length}(r)\}$ such that for all $n \leq \min\{\mathbf{length}(t), \mathbf{length}(r)\}$, $(t)_{m+n} = (r)_{k+n}$, then we shall call t and r matching transitions. \square

Matching transitions are precisely the *composable transitions*, i.e. precisely those pairs of transitions in the two nets that can be *composed*, allowing for output from one net to flow, as input, in the other net. In the next Section, we use this notion of matching transitions to define an operator of composition of open nets.

3 Operators on Open Nets

In this section we define some natural operators on oPN's, including a 0-ary `nil` operator, three different types of prefix operators, composition and choice. We start with an auxiliary and elementary operation of “place fusion”, which we shall need in order to provide rigorous definitions for composition and for the solution of recursive equations on nets.

3.1 Place Fusion

If \mathcal{A} is an oPN and A, B are place labels, possibly labeling places of \mathcal{A} , appropriately identifying A, B in \mathcal{A} results in a new oPN $\mathcal{A}_{/A \leftarrow B}$. This operation is of interest only in the context of defining the composition operation (as well as in computing fixed points).

Technically, the renaming operation (place fusion) $/_{B \leftarrow A}$ is a unary operator with $\mathcal{A}_{/B \leftarrow A}$ being the net obtained by renaming all places or ports labeled by B , using the new label A (which may already be used to label a place or port in the net) and appropriately modifying the transition relation and marking of the net. The net $\mathcal{A}_{/B \leftarrow A}$ is rigorously defined as follows:

- The places of $\mathcal{A}_{/B \leftarrow A}$ are exactly $(Pl(\mathcal{A}) \setminus \{B\}) \cup \{A\}$
- $M_{0, \mathcal{A}_{/B \leftarrow A}} = \begin{cases} M_{0, \mathcal{A}} & \text{if } B \notin M_{0, \mathcal{A}} \\ M_{0, \mathcal{A}} \setminus \{B\} \cup \{A\} & \text{otherwise} \end{cases}$
- The transition relation is derived from that of \mathcal{A} , in the obvious way.
 - If $I \xrightarrow{u} O$ is a transition in \mathcal{A} and $(I \cup O) \cap \{B\} = \emptyset$, then $I \xrightarrow{u} O$ is a transition in $\mathcal{A}_{/B \leftarrow A}$
 - If $B \notin I \xrightarrow{u} O \cup \{B\}$ is a transition in \mathcal{A} , then $I \xrightarrow{u} O \cup \{A\}$ is a transition in $\mathcal{A}_{/B \leftarrow A}$
 - If $I \cup \{B\} \xrightarrow{u} O \not\supseteq B$ is a transition in \mathcal{A} , then $I \cup \{A\} \xrightarrow{u} O$ is a transition in $\mathcal{A}_{/B \leftarrow A}$
 - If $I \cup \{B\} \xrightarrow{u} O \cup \{B\}$ is a transition in \mathcal{A} , then $I \cup \{A\} \xrightarrow{u} O \cup \{A\}$ is a transition in $\mathcal{A}_{/B \leftarrow A}$.

The operation generalizes to more than one equations in the obvious way:

$$\mathcal{A}_{/B \leftarrow A, D \leftarrow C} = (\mathcal{A}_{/B \leftarrow A})_{/D \leftarrow C}$$

and similarly for more equations.

3.2 nil

A 0-ary operator; it designates an oPN consisting of a single, ordinary place. Formally, for any place A , **nil** is defined as the oPN $(\{A\}, \emptyset, \{A\})$ (the choice of the place name A is of no significance when considering behaviorally equivalent nets, as we shall eventually do).

3.3 prefix

A unary operator; it comes in three variants: $\tau_*(\cdot)$, $c!_*(\cdot)$ and $c?_*(\cdot)$.

Definition 7 (Prefix). *Let*

$$\mathcal{A} = (P, \longrightarrow, M_0)$$

be an oPN and $A, B, x \notin P$ with $A, B \in \text{Place}$ and $x \in \text{Port}$.

– $\tau_*\mathcal{A}$ is the net

$$(P \cup \{A\}, \longrightarrow \cup \{\{A\} \xrightarrow{\tau} M_{0,\mathcal{A}}\}, \{A\})$$

– $c!_*\mathcal{A}$ is the net

$$(P \cup \{A, x\}, \longrightarrow_{c!}, \{A\})$$

where $\longrightarrow_{c!}$ is defined by:

$$\longrightarrow_{c!} = \longrightarrow \cup \{\{A\} \xrightarrow{c!} M_{0,\mathcal{A}} \cup \{x\}\}$$

– $c?_*\mathcal{A}$ is the net

$$(P \cup \{A, x\}, \longrightarrow_{c?}, \{A\})$$

where $\longrightarrow_{c?}$ is defined by:

$$\longrightarrow_{c?} = \longrightarrow \cup \{\{A\} \xrightarrow{c?} \{B\}, \{B, x\} \xrightarrow{\tau} M_{0,\mathcal{A}}\}. \quad \square$$

Since eventually we shall only be interested in open nets up to a natural notion of behavioral equivalence, the choice of the place and port names A, B, x in the above definitions is immaterial.

3.4 Choice

The choice operator \oplus is a binary operator, where the net $\mathcal{A} \oplus \mathcal{B}$ is formed by adding a new place name A and two transitions $\{A\} \xrightarrow{\tau} M_{0,\mathcal{A}}$ and $\{A\} \xrightarrow{\tau} M_{0,\mathcal{B}}$.

Definition 8 (Summation). *Let*

$$\mathcal{A} = (P_1, \longrightarrow_1, M_{0,1}), \mathcal{B} = (P_2, \longrightarrow_2, M_{0,2})$$

be two oPNs and $A \in \text{Place not in } P_1 \cup P_2$. Then

$\mathcal{A} \oplus \mathcal{B}$ is the net

$$(P_1 \cup P_2 \cup \{A\}, \longrightarrow_{\oplus}, \{A\})$$

where \longrightarrow_{\oplus} is defined by:

$$\longrightarrow_{\oplus} = \longrightarrow_1 \cup \longrightarrow_2 \cup \{\{A\} \xrightarrow{\tau} M_{0,1}, \{A\} \xrightarrow{\tau} M_{0,2}\}.$$

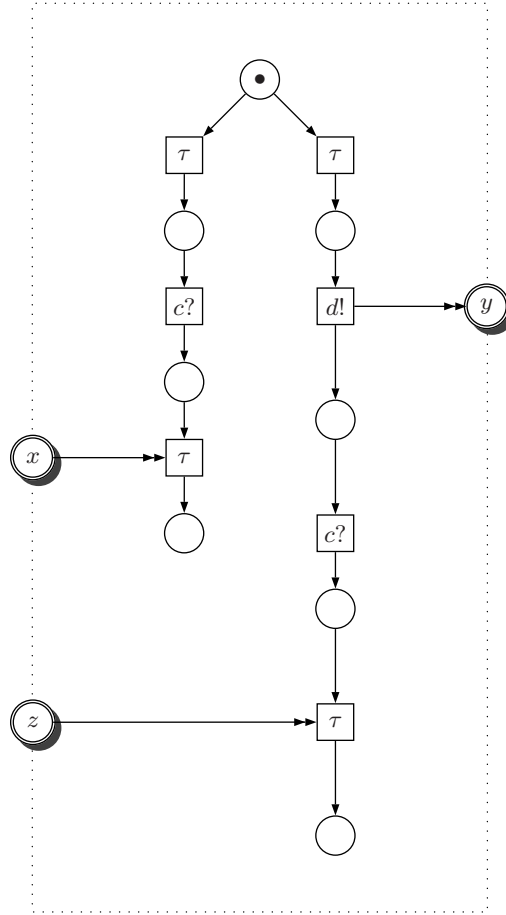
In Figure 1 we show an example of these operators, namely the net $c?_*\mathbf{nil} \oplus d!_*c?_*\mathbf{nil}$.

3.5 Concurrent Composition of Open Petri Nets

We introduced this operation in [4], from which we briefly review the definition.

To make the concept simple to understand, we first define a notion of composition at a single pair of matching transitions.

Fig. 1. The net $c?_{*}nil \oplus d!_{*}c?_{*}nil$

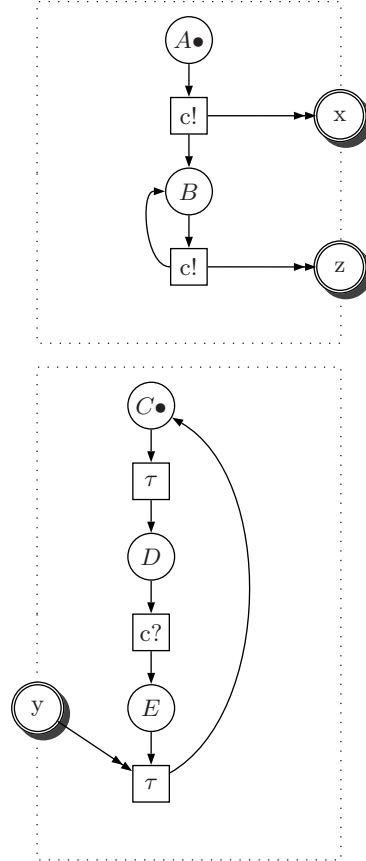


Definition 9. Let \mathcal{A}, \mathcal{B} be two nets, $t = I \xrightarrow{c!}_A O \cup \{x\}$ an output transition in \mathcal{A} and $r = J \xrightarrow{c?}_B K, K \cup \{y\} \xrightarrow{\tau}_B L$ a matching input prompt transition in \mathcal{B} (we do not assume that $x \neq y$). Assume the set of place (but not necessarily of port) names of \mathcal{A} is disjoint from the corresponding set of \mathcal{B} (otherwise, injectively rename them). Let also z be a new port name (not occurring in either \mathcal{A} or \mathcal{B}). The join $\mathcal{A} \otimes_{t,r} \mathcal{B}$ of the nets at the matching pair (t, r) is the net obtained as the union of \mathcal{A}, \mathcal{B} , after renaming both ports x (in \mathcal{A}) and y (in \mathcal{B}) to z . In other words, let $\mathcal{A}' = \mathcal{A}_{/x \leftarrow z}$, $\mathcal{B}' = \mathcal{B}_{/y \leftarrow z}$ and $\mathcal{A} \otimes_{t,r} \mathcal{B} = \mathcal{A}' \cup \mathcal{B}' = \mathcal{A}_{/x \leftarrow z} \cup \mathcal{B}_{/y \leftarrow z}$. \square

This operation can be generalized to any number of matching transitions. We first present an example.

Example 1. Consider the two nets in Figure 2.

Fig. 2. Example for the join of two nets



The net \mathcal{A} on top performs an initial output transition on channel c and then gets into a loop, continuously performing output on c .

Let $t_1 = \{A\} \xrightarrow{c!} \{x, B\}$ and $t_2 = \{B\} \xrightarrow{c!} \{z, B\}$ be the two transitions of \mathcal{A} .

The net \mathcal{B} at the bottom executes a silent action, prompts for input and, if input is received, it consumes it and repeats this behavior.

Let $r = \{D\} \xrightarrow{c?} \{E\}$ be the input transition of \mathcal{B} .

Then we have $\text{length}(t_1) = 1$, $(t_1)_1 = 1$, $\text{length}(t_2) = \omega$, $(t_2)_k = k + 1$, $\text{length}(r) = \omega$ and $(r)_k = k$.

Notice that

$$(t_1)_1 = 1 = (r)_1$$

$$(t_2)_k = k + 1 = (r)_{k+1}$$

By Definition 6 both (t_1, r) and (t_2, r) are composable pairs of transitions. To obtain the desired composition, let z_1, z_2 be two new port names. Let

$$\mathcal{A}^{(1)} = \mathcal{A}_{/x \leftarrow z_1} \quad \mathcal{B}^{(1)} = \mathcal{B}_{/y \leftarrow z_1}$$

$$\mathcal{A}^{(2)} = \mathcal{A}_{/z_1 \leftarrow z_2} \quad \mathcal{B}^{(2)} = \mathcal{B}_{/z_1 \leftarrow z_2}$$

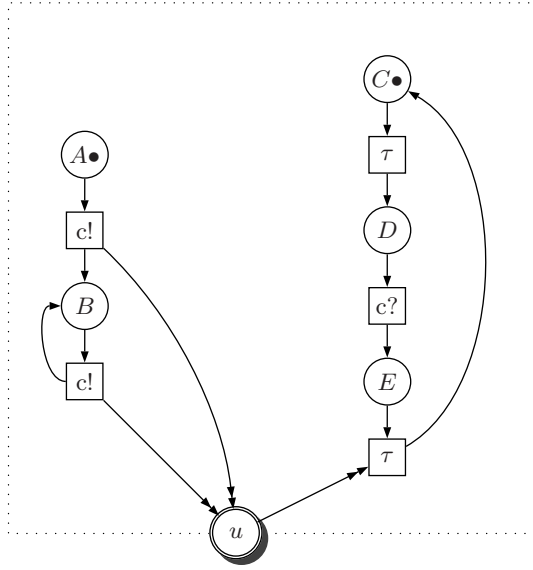
and

$$\mathcal{A} \otimes \mathcal{B} = \mathcal{A}^{(2)} \cup \mathcal{B}^{(2)}$$

$$= \mathcal{A}_{/x \leftarrow z_1, z_1 \leftarrow z_2} \cup \mathcal{B}_{/y \leftarrow z_1, z_1 \leftarrow z_2}$$

The resulting composite net is shown in Figure 3. The reader may wish to ex-

Fig. 3. Concurrent join of the nets of Figure 2 (without hiding the communication port)



periment with variations of the example, with more than one initial $c!$ moves on the left, before getting into the loop, and with several $c?$ transitions on the right, before getting into the input loop as well. \square

Having illustrated, with an example, the intended general notion of composition of open petri nets, we proceed to the formal definition, after a small technical Lemma.

Lemma 1. *Let \mathcal{A}, \mathcal{B} be two nets and J be the set of all matching pairs of transitions (transitions t of \mathcal{A} , paired with matching transitions r in \mathcal{B}). There exists a renaming function σ , defined on the port labels of \mathcal{A}, \mathcal{B} , such that in $\mathcal{A}\sigma$, $\mathcal{B}\sigma$ all matching pairs in J have their associated communication ports tagged by the same label.*

Proof. We have presented the proof in [4], but it is useful to repeat it here.

If J is finite, then obvious. Otherwise, let j_0, \dots, j_n, \dots be an enumeration of the members of J and for each j_i , let (x_i, y_i) be the pair of port names of the two matching transitions in j_i . Let also $(z_i)_i$ be a sequence of new port names (not occurring in either \mathcal{A} or \mathcal{B}). Define a sequence of partial renaming functions as follows:

σ_0 : $\text{dom}(\sigma_0) = \{x_0, y_0\}$ and $\sigma_0(x_0) = z_0 = \sigma_0(y_0)$
 σ_{n+1} : Having defined σ_n , let
 $\text{dom}(\sigma_{n+1}) = \text{dom}(\sigma_n) \cup \{x_n, y_n\}$
Case $x_n, y_n \notin \text{dom}(\sigma_n)$
Then extend σ_n , setting $\sigma_{n+1}(x_n) = z_n = \sigma_{n+1}(y_n)$
Case $\{x_n, y_n\} \subseteq \text{dom}(\sigma_n)$
Then let $\sigma_{n+1} = \sigma_n$
Case $x_n \in \text{dom}(\sigma_n)$, but $y_n \notin \text{dom}(\sigma_n)$
Let $\sigma_{n+1}(x_n) = \sigma_n(x_n) = \sigma_{n+1}(y_n)$
Case $x_n \notin \text{dom}(\sigma_n)$, but $y_n \in \text{dom}(\sigma_n)$
Let $\sigma_{n+1}(x_n) = \sigma_n(y_n) = \sigma_{n+1}(y_n)$
 σ : Define $\sigma = \bigcup_n \sigma_n$

By definition of σ , if $(t, r) = j_k$ is the k -th matching pair, then in $\mathcal{A}\sigma$ and $\mathcal{B}\sigma$ the matching transitions t, r have their corresponding ports tagged by the same label.

Definition 10 (External Concurrent Join of Open Nets). *Let \mathcal{A}, \mathcal{B} be two nets, assuming their sets of ports and places are disjoint (if not, injectively rename each one of them) and let σ be the port renaming function of matching transitions, as in Lemma 1. The external concurrent join of \mathcal{A}, \mathcal{B} , is defined as*

$$\mathcal{A} \otimes \mathcal{B} = \mathcal{A}\sigma \cup \mathcal{B}\sigma$$

where recall that $\mathcal{A}\sigma$ is the net obtained from \mathcal{A} , but with communication port names modified by σ . \square

Remark 1 (Internal and Mixed Join). We can slightly modify our definitions and obtain notions of *internal concurrent join* (hiding the ports after composition) and *mixed concurrent join* (selectively hiding or not the communication ports, depending on the nature of the communication to take place).

To achieve this, the definition of oPN's needs to be slightly modified, allowing for a distinction between *external* and *internal* communication ports, Port_{int} , Port_{ext} , so that internalization of all (or some) ports after composition can be

performed. But also, by introducing a distinction between private and public (internal and external) communicative actions.

The case of internalizing all ports after composition of matching transitions is straightforward, by letting $\mathcal{A}|\sigma$ (where σ is the renaming function of Lemma 1) be the net obtained by first renaming according to σ , but also by removing from its set of external communication ports the range of σ and adding it to its set of internal communication ports.

For the case of a mixed concurrent join, communicative actions can be assigned an annotation, e.g. $!c : \text{in}$, $?d : \text{ex}$ (or no annotation, meaning that both private and public communication are acceptable). This affects the definition of matching (composable) transitions, since now the additional requirement that the annotations agree must be imposed. Furthermore, the set of composable transitions for two oPNs splits into two sets J_{in}, J_{ex} and similarly for the renaming function σ of Lemma 1, splitting to σ_{in}, σ_{ex} . The mixed concurrent join can be then simply defined as $\mathcal{A}\sigma_{ex}|\sigma_{in} \cup \mathcal{B}\sigma_{ex}|\sigma_{in}$. Details are left to the interested reader. \square

4 Polynomial Nets and Equations

In this Section we deal with recursion (cyclic nets, specified with net terms of the form $\mu X.\mathcal{P}$). To do this, we proceed as in our [3], solving equations in a class of *polynomial nets*.

The class \mathcal{N} of open petri nets, together with the operators we defined,

$$\langle \mathcal{N}, \text{nil}, \tau_*, c!_*, c?_*, \oplus, \otimes \rangle$$

is an algebraic structure and we may consider *polynomials* over that structure. To do this, we let \mathcal{X} be a set of *indeterminates* disjoint from the sets of place and port names of oPNs in \mathcal{N} . The class $\mathcal{N}[\mathcal{X}]$ of *polynomial open nets* in the indeterminates in \mathcal{X} is the smallest class satisfying the conditions

- If \mathcal{A} is an oPN (i.e. $\mathcal{A} \in \mathcal{N}$), then $\mathcal{A} \in \mathcal{N}[\mathcal{X}]$
- If X is an indeterminate, then $\langle \{X\}, \emptyset, \{X\} \rangle \in \mathcal{N}[\mathcal{X}]$
- If $\mathcal{A}, \mathcal{B} \in \mathcal{N}[\mathcal{X}]$, then
 - $\tau_*\mathcal{A}, c!_*\mathcal{A}, c?_*\mathcal{A} \in \mathcal{N}[\mathcal{X}]$
 - $\mathcal{A} \oplus \mathcal{B}, \mathcal{A} \otimes \mathcal{B} \in \mathcal{N}[\mathcal{X}]$

It is convenient to introduce a syntax for members of $\mathcal{N}[\mathcal{X}]$, letting members of \mathcal{N} name themselves. In the syntax below, see (2), all but the last term have the obvious interpretation as polynomial open nets, given the definition of the operators we have provided, and we shall not make any notational distinction between the terms and their interpretation.

Our claim is that the class \mathcal{N} also contains nets that satisfy equations of the form $X = \mathcal{P}(X)$. A solution to such an equation is designated in the syntax below by the term $\mu X.\mathcal{P}$.

$$\begin{aligned} \mathcal{P} := & \mathcal{A} (\mathcal{A} \in \mathcal{N}) \mid X (X \in \mathcal{X}) \mid \text{nil} \mid \tau_*\mathcal{P} \mid c!_*\mathcal{P} \mid c?_*\mathcal{P} \mid \\ & \mid \mathcal{P} \oplus \mathcal{P} \mid \mathcal{P} \otimes \mathcal{P} \mid \mu X.\mathcal{P} \end{aligned} \tag{1}$$

In the syntax above, for recursively specified nets $\mu X.\mathcal{P}$, the indeterminate X is assumed to be *guarded* in the recursion body \mathcal{P} (i.e. it appears within the scope of a prefix operator). As mentioned above, a recursively specified net $\mu X.\mathcal{P}$ designates a solution of the equation $X = \mathcal{P}$. We will argue for the existence of solutions, but we first note that, because of nested recursions, a recursively specified net $\mu X.\mathcal{P}$ is in fact equivalently specified as the solution of a system of equations (soe, in the sequel)

$$(X_i = \mathcal{P}_i(X_1, \dots, X_n))_{i=1, \dots, n}$$

For example, if $\mathcal{P} = \mu X.(a_*X \oplus \mu Y.(b_*Y \oplus X))$, where a, b are action names, then because of the recursively specified subnet $\mu Y.b_*Y \oplus X$, the associated soe is

$$\begin{aligned} X &= a_*X \oplus Y \\ Y &= b_*Y \oplus X \end{aligned}$$

As a matter of agreement, given a system of polynomial equations, we will always list first the equation corresponding to the top recursive term. Consider the soe $\bar{e} = \{e_1, \dots, e_n\}$ below:

$$\begin{aligned} (e_1) \quad X_1 &= \mathcal{P}_1(X_1, \dots, X_n) \\ (e_2) \quad X_2 &= \mathcal{P}_2(X_1, \dots, X_n) \\ &\vdots \\ (e_n) \quad X_n &= \mathcal{P}_n(X_1, \dots, X_n) \end{aligned}$$

Each one of $\mathcal{P}_1, \dots, \mathcal{P}_n$ is a net where the indeterminates amongst X_1, \dots, X_n that appear in \mathcal{P}_i label terminal places.

To solve the soe we proceed by induction on the depth $n \geq 1$ of $\mathcal{P} = \mu X_1.\mathcal{P}_1(X_1, \dots, X_n)$.

If $n = 1$, then \mathcal{P}_1 has no recursive subterms and the soe reduces (omitting the subscript) to a single equation $X = \mathcal{P}(X)$ in $\mathcal{N}[X]$. In fact, it suffices to consider this case only since, if $n > 1$, then by induction each of the equations $X_{i+1} = \mathcal{P}_{i+1}(X_1, \dots, X_n)$ has a solution $\mathcal{A}_{i+1} \in \mathcal{N}[X_1]$. Then the first equation becomes $X_1 = \mathcal{P}_1(X_1, \mathcal{A}_2, \dots, \mathcal{A}_n)$ and thus the problem reduces to a single equation in $\mathcal{N}[X_1]$, to be solved in \mathcal{N} .

Before proceeding to constructing solutions to equations, we present a simple example.

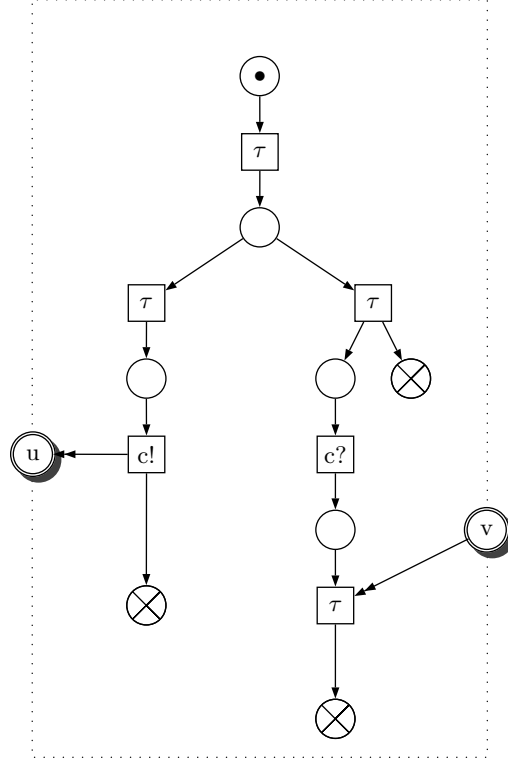
Example 2. The net in Figure 4 is the polynomial net corresponding to $\mathcal{P}(X) = \tau_*(c!_*X \oplus (X \otimes c?_*X))$. In drawing the polynomial net, we have drawn instances of the indeterminate X as a store, for emphasis. Note that we speak of “instances of the indeterminate X ” because of renaming taking place in the case of the concurrent join and choice.

The net in Figure 5 is the solution of the equation

$$X = \tau_*(c!_*X \oplus (X \otimes c?_*X))$$

designated by the term $\mu X.\tau_*(c!_*X \oplus (X \otimes c?_*X))$. □

Fig. 4. The polynomial net $\mathcal{P}(X) = \tau_*(c!_*X \oplus (X \otimes c?_*X))$. For clarity, we draw all instances of the indeterminate X using the store notation.

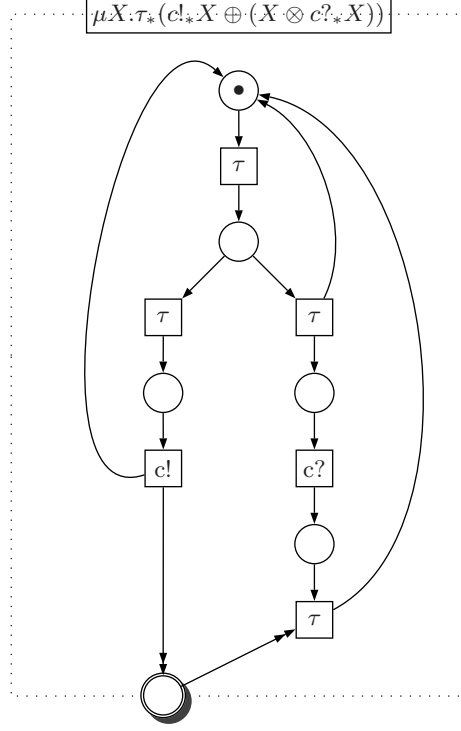


Remark 2. In [3], some of the operators (in particular, the choice and synchronous product operations) did not allow to resolve terms involving the indeterminate as a component (e.g. terms like $X \otimes \mathcal{A}$, or $X \oplus \mathcal{A}$). In the present report we dealt with *internal* choice, which gets resolved to its components by an internal τ move. Also, the product operation (concurrent join) is essentially a union operation (with possible sharing of a communication port). As a result, in any polynomial net of the form $\mathcal{P}(X)$, arising according to (2), X will only appear as a name of a terminal place. This is manifested in Figure 4. As a consequence, and unlike the situation in our [3], capturing the recursive structure appears to be elementary in the present context, due to the nature of the operators considered here. However, a novel complication arises, when considering communication ports. \square

Theorem 1. *Every equation $X = \mathcal{P}(X)$ in $\mathcal{N}[X]$ has a solution in \mathcal{N} .*

Proof. To solve an equation of the form $X = \mathcal{P}(X)$ we proceed as follows:

Fig. 5. The solution to the equation $X = \tau_*(c!_*X \oplus (X \otimes c?_*X))$



- First, replace each transition $t = I \xrightarrow{a} O \cup \{X\}$ in $\mathcal{P}(X)$ by the transition $t' = I \xrightarrow{a} O \cup M_0$ and, if $\mathcal{P}(X) = \langle P, T, M \rangle$ and T' is the resulting (modified) transition relation, then let $\mathcal{P}_1 = \langle P \setminus \{X\}, T', M \rangle$. Figure 6 shows the net \mathcal{P}_1 for the specific example of the net in Figure 4.
- Second, rank each subterm of $\mathcal{P}(X)$ according to the number of occurrences of the product operator \otimes . Next, define a port renaming function σ , by induction on the **rank** of subterms of $\mathcal{P}(X)$.
 - Let σ_0 (corresponding to subterms $\mathcal{Q}_{i,0}$ of **rank** 0) be the empty function.
 - Next, define σ_1 .
Let $\mathcal{Q}_{j,1} = \mathcal{A}_{j1}(X) \otimes \mathcal{A}_{j2}(X)$, for $j = 1, \dots, k$, be all the product subterms of $\mathcal{P}(X)$ of **rank** 1 (for an illustration, this is the case of the term $X \otimes c?_*X$ in Example 2, see also Figure 4).
Consider the terms $\mathcal{A}_{ji}(\mathcal{P}_1)$, $i = 1, 2$ (in the example, these are the terms \mathcal{P}_1 and $c?\mathcal{P}_1$). Let $\sigma_{j,1}$, for each j , be the port renaming of Lemma 1, used in forming the product $\mathcal{A}_{j1}(\mathcal{P}_1) \otimes \mathcal{A}_{j2}(\mathcal{P}_1)$.
Define σ_1 in k steps, as follows: Let $\sigma_1^1 = \sigma_{1,1}$. Having defined σ_1^r and while $r < k$, for each $x \in \text{dom}(\sigma_1^r) \cap \text{dom}(\sigma_{r+1,1})$ modify $\sigma_{r+1,1}$ so as to agree with σ_1^r on x . More specifically, if $\sigma_{r+1,1}(x) = z$, let $A = \sigma_{r+1,1}^{-1}(z)$

and define

$$\sigma'_{r+1,1}(u) = \begin{cases} \sigma_{r+1,1}(u) & \text{If } u \in \text{dom}(\sigma_{r+1,1}) \setminus A \\ \sigma_1^r(u) & \text{otherwise} \end{cases}$$

Do this for each $x \in \text{dom}(\sigma_1^r) \cap \text{dom}(\sigma_{r+1,1})$, let $\sigma'_{r+1,1}$ be the modified function and let $\sigma_1^{r+1} = \sigma_1^r \cup \sigma'_{r+1,1}$. Finally, set $\sigma_1 = \bigcup_r \sigma_1^r$.

- Having defined σ_n and while $n < \text{rank}(\mathcal{P}(X))$, let $\mathcal{Q}_{t,n+1} = \mathcal{A}_{t1}(X) \otimes \mathcal{A}_{t2}(X)$, for $t = 1, \dots, s$, be all the product subterms of $\mathcal{P}(X)$ of rank $n + 1$.

Consider the terms $\mathcal{A}_{ti}(\mathcal{P}_1\sigma_n)$, $i = 1, 2$ (recall that $\mathcal{P}\sigma$ denotes the net obtained from \mathcal{P} after renaming ports using σ). Let $\sigma_{t,n+1}$, for each t , be the port renaming of Lemma 1, used in forming the product $\mathcal{A}_{t1}(\mathcal{P}_1\sigma_n) \otimes \mathcal{A}_{t2}(\mathcal{P}_1\sigma_n)$.

Define σ_{n+1} in s steps, as follows: First, for each $x \in \text{dom}(\sigma_n) \cap \text{dom}(\sigma_{1,n+1})$, modify $\sigma_{1,n+1}$ to $\sigma'_{1,n+1}$ so as to agree with σ_n (in the same way as explained in the previous step) and set $\sigma_{n+1}^1 = \sigma_n \cup \sigma'_{1,n+1}$.

Having defined σ_{n+1}^p and while $p < s$, for each $x \in \text{dom}(\sigma_{n+1}^p) \cap \text{dom}(\sigma_{p+1,n+1})$ again modify $\sigma_{p+1,n+1}$ to $\sigma'_{p+1,n+1}$ so as to agree with σ_{n+1}^p on x (again, in the same way as explained above) and set $\sigma_{n+1}^{p+1} = \sigma_{n+1}^p \cup \sigma'_{p+1,n+1}$. Finally, set $\sigma_{n+1} = \bigcup_p \sigma_{n+1}^p$.

Having defined an increasing sequence of port renamings

$$\sigma_1 \subseteq \sigma_2 \subseteq \dots \subseteq \sigma_n \subseteq$$

for $n \leq \text{rank}(\mathcal{P}(X))$, let $\sigma = \bigcup_n \sigma_n$.

The solution to the equation $X = \mathcal{P}(X)$ can be now defined by $\mu X. \mathcal{P} = \mathcal{P}_1\sigma$. Indeed, \mathcal{P}_1 captures the recursive behavior, while, by its construction, σ performs all necessary port identifications that become necessary when passing from $\mathcal{P}(X)$ to \mathcal{P}_1 . In the example we presented, the resulting solution is demonstrated in Figure 5, as the reader may easily verify.

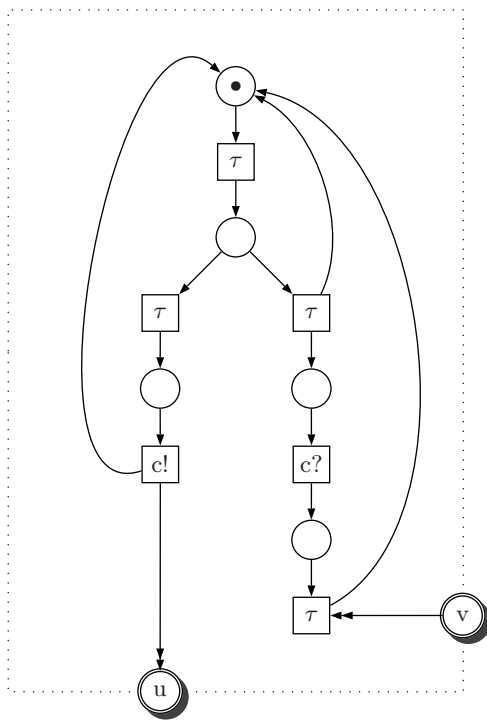
5 Conclusions and Further Research

In Section 4 we introduced a syntax, where each term of the syntax has a straightforward interpretation as a polynomial petri net. As the reader has no doubt noticed, this is essentially the syntax of value-passing CCS.

$$\begin{aligned} \mathcal{P} := & X \ (X \in \mathcal{X}) \mid \mathbf{nil} \mid \tau.\mathcal{P} \mid c![v]\mathcal{P} \mid c?(x)\mathcal{P} \mid \\ & \mid \mathcal{P} \oplus \mathcal{P} \mid \mathcal{P} \otimes \mathcal{P} \mid \mu X.\mathcal{P} \end{aligned} \quad (2)$$

where $c![v]\mathcal{P}$ is the process that can output the value v on channel c and continue like \mathcal{P} and $c?(x)\mathcal{P}$ is the process that prompts for input on c and, when an input value v is received it behaves like the functional application $(\lambda x.P)v$, i.e. like $P[v/x]$. The only difference is that in the present setting we focused on

Fig. 6. The net \mathcal{P}_1 of the proof of Theorem 1



communication per se and have only admitted a single value (a token) to be passed around during communication. However, we do not expect any significant issues, or difficulties, to be raised by allowing multiple values of distinct basic types to be passed around during communication, except for needing to distinguish different *colors* of tokens in our petri net setting.

We have established, in this report, a framework for the modeling and analysis of communicating services, with message passing, which is semantically based on petri nets, while at the same time the language for designating open nets is a standard process algebraic language. This makes it possible, we think, to employ both process algebraic and petri net based techniques in the study of communicating services.

From a process algebraic viewpoint, it should be interesting to extend our present results, by aiming to establish a full abstraction result. It should be also interesting to extend in the direction of higher-order value passing, where processes themselves are being transmitted as values.

Finally, we pointed out in Remark 1 that our setting allows for both an internal and an external operation of concurrent join, as well as a mixed approach.

It should be then interesting to further investigate the possibility to extend our present results in the direction of providing semantics for the π -calculus.

We will leave such extensions, however, for subsequent research.

References

1. W.M.P. van der Aalst, "A class of petri nets for modeling and analysing business processes", Computing science report 95/26, Eindhoven University of Technology, 1995.
2. F. Leymann, D. Roller, and M. Schmidt. Web services and business process management. *IBM Systems Journal*, 41(2), 2002.
3. Hartonas, C., Kodokostas, D. and Kokkinos, K., "Petri Net Semantics for Communicating Agents", in *Annals of Mathematics, Computing & Teleinformatics*, vol 1-4, pp 31-40, 2006.
4. Hartonas, C., "Modeling Service Communication with Open Petri Nets", Technical Report, TEI Larissa, 2007.
5. A. Martens. *Verteilte Geschäftsprozesse - Modellierung und Verifikation mit Hilfe von Web Services*. PhD thesis, Institut für Informatik, Humboldt-Universität zu Berlin, 2004.
6. Axel Martens, "On Compatibility of Web Services", <http://www2.informatik.hu-berlin.de/top/download/publications/WSCompatibility.pdf>
7. Massuthe, P., Reisig, W. and Schmidt, K., "An Operating Guideline Approach to the Service Oriented Architecture", in *Annals of Mathematics, Computing & Teleinformatics*, vol 1-3, pp 35-44, 2005.
8. Kathrin Kaschner, Peter Massuthe, and Karsten Wolf, "Symbolic Representation of Operating Guidelines for Services", *Petri Net Newsletter*, 72:21-28, April 2007.
9. Twan Basten, *In Terms of Nets: System Design with Petri Nets and Process Algebra*, PhD thesis, Eindhoven University of Technology, 1998 (ISBN 90-386-0631-1).