Towards Petri Net Calculi based on Synchronization via Places*

Nikolay A. Anisimov *and* Alexey A. Kovalenko

Institute for Automation and Control Processes
Far East Branch of the Russian Academy of Sciences
5 Radio Str., Vladivostok, 690041, Russia

Abstract

This paper addresses the problem of designing Petri net based calculi. It is pointed out that almost all existing problem–oriented Petri net calculi have been developed in an ad hoc fashion, and the need for a basic formal tool which will help their design is stressed. We introduce a series of place synchronization operations ranging from primitive place synchronization to the general synchronization via place access points. We consider some examples where the place synchronization operation can be successfully used to model place merging, sequentialization and disabling.

Key words and phrases: Petri nets, compositionality, synchronization, access point, disabling.

1 Introduction

It is well–recognized that Petri nets and related models are very useful and powerful formalisms for the description and verification of concurrent and distributed systems. At the same time, it is admitted that Petri nets lack compositionality and modularity, which prevents them from being widely applied to real–world systems of industrial size. There has been a number of attempts aimed at bringing compositionality and modularity into Petri nets. Roughly speaking, all these efforts involve the design of Petri net based calculi which would enable one to construct complex nets from less complex components and to predict their properties.

We can distinguish two main streams of work in this area. The first one was inspired by process calculi like CCS [11]. It comprises papers dealing with process calculi (e.g. [7, 12]) and languages (OCCAM [8] and LOTOS [10]). The second stream is concerned with the building of Petri net calculi in their own right without referring to any specific language (e.g. [4, 9]).

The existing net calculi are not entirely satisfactory. The main reason for this is that they are always very specialized, i.e. they have been developed for a specific language (process calculus) or an area of application. Another drawback results from the fact that they have been designed in an *ad–hoc* fashion. As a result, the net operations used are very lengthy and technically difficult, while their intuitive meaning is quite clear.

In this paper we advocate an approach in which we propose a small set of basic operations, which may have technically complex definitions. At the same time, however, using these operations, we can easily define the set of main operations for any problem–oriented Petri net calculus. Moreover, using the properties of these basic operations, we can easily derive the properties of the derived operations.

In our approach, we distinguish two basic net operations, called *transition* and *place synchronization*, which seem to be quite orthogonal to each other. In this paper we study the operation of place synchronization. We introduce some increasingly complex definitions of synchronization which finally lead us to the most general one – ***a general place synchronization*** based on the notion of ***a place access point***.

2 Basic notions

In this section we recall some notions and definitions from the Petri net theory. Note that the definitions are slightly different from the standard ones.

The powerset of a set A will be denoted as $\mathcal{P}(A)$. For $\rho \subseteq \mathcal{P}(A)$, we denote $\|\rho\| = \bigcup_{x \in \rho} x$.

Definition 2.1 *Let K be a set called the transition alphabet. A net over the alphabet K is $N = \langle S, T \rangle$ where*

*This work is supported by the Russian Basic Research Fund (Project 93-013-17372).

0-8186-7038-X/95 $4.00 © 1995 IEEE

Proceedings of the First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis (PAS '95)
0-8186-7038-X/95 $10.00 © 1995 IEEE

1. $S = \{s_1, s_2, ..., s_n\}$ *is a set of places;*

2. $T \subseteq \mathcal{P}(S) \times K \times \mathcal{P}(S)$ *is a set of transitions.*

Notation 2.2 *Let* $N = \langle S, T\rangle$ *be a net over* K *and* $s \in S$, $t \in T$, $T' \subseteq T$, $S' \subseteq S$.

1. *If* $t = \langle Q_1, a, Q_2\rangle \in T$, *then* ${}^\bullet t = Q_1$, $t^\bullet = Q_2$, *and* $\underline{t} = a$. *The sets* ${}^\bullet t$ *and* $t^\bullet$ *are respectively called pre-set and post-set of* t*; and* $\underline{t}$ *is called the name of the transition. I.e.* $t = \langle {}^\bullet t, \underline{t}, t^\bullet\rangle$*;*

2. ${}^\bullet(T') = \bigcup_{t\in T'} {}^\bullet t$, $(T')^\bullet = \bigcup_{t\in T'} t^\bullet$, ${}^\bullet(T')^\bullet = {}^\bullet(T') \cup (T')^\bullet$*;*

3. ${}^\bullet s = \{t \mid s \in t^\bullet\}$, $s^\bullet = \{t \mid s \in {}^\bullet t\}$*;*

4. ${}^\bullet(S') = \bigcup_{s\in S'} {}^\bullet s$, $(S')^\bullet = \bigcup_{s\in S'} s^\bullet$, ${}^\bullet(S')^\bullet = {}^\bullet(S') \cup (S')^\bullet$.

We use the standard graphical representation of nets in which places are represented as circles, and transitions as boxes with directed arcs. The names of transitions will be placed inside the appropriate boxes.

3 Place synchronization

Let Δ be a finite set of names called *an alphabet*, and $\overline{\Delta} = \{\bar{a} \mid a \in \Delta\}$ be the associated set of complement names or co-names. In other words, we define a bijection $\bar{\cdot} : \Delta \to \overline{\Delta}$, which defines a one-to-one correspondence between each name and its co-name. For convenience, the inverse of $\bar{\cdot}$ is also denoted as $\bar{\cdot}$. Thus we have $\bar{\bar{a}} = a$. The function $\bar{\cdot}$ can be extended in obvious way to sets $X \subseteq \Delta \cup \overline{\Delta}$. For example, if $X = \{a, \bar{a}, b, c, \bar{e}\}$, then $\overline{X} = \{\bar{a}, a, \bar{b}, \bar{c}, e\}$. Let $\tau \notin \Delta \cup \overline{\Delta}$ be a distinguished symbol that is usually associated with an unobservable action. We will denote $\Lambda_\Delta = \Delta \cup \overline{\Delta}$ and $Act_\Delta = \Lambda_\Delta \cup \{\tau\}$.

Definition 3.1 *Let* $N = \langle S, T\rangle$ *be a net,* Δ *an alphabet, and* a *a name.*

1. *A primitive a-labelling of the net* N *is a function* $\pi_a : S \to \{a, \bar{a}, \tau\}$*;*

2. *A simple labelling of the net* N *over* Δ *is a function* $\pi_\Delta : S \to Act_\Delta$.

In a primitive labelling, each place in N can be labelled by a or $\bar{a}$ or τ. In the last case it is considered to be invisible w.r.t. labelling π_a. Sometimes, when it does not lead to a confusion, we will write π and Act instead of π_Δ and Act_Δ. In the following, we will use a function $Alph : \mathcal{P}(Act_\Delta) \to \mathcal{P}(\Delta)$ defined as follows:

$$Alph(\tau) = \emptyset,\ Alph(a) = Alph(\bar{a}) = \{a\},$$

$$Alph(X) = \bigcup_{x\in X} Alph(x)$$

We also will use: $Alph(\pi) = \bigcup_{(s,x)\in\pi} Alph(x)$. Clearly, $Alph(\pi_a) = \{a\}$.

Notation 3.2 *Let* $N = \langle S, T\rangle$ *be a net and* π_a *its labelling.*

1. $S^a = \{s \in S \mid \pi_a(s) = a\}$,
$S^{\bar{a}} = \{s \in S \mid \pi_a(s) = \bar{a}\}$, $S^{a\bar{a}} = S^a \cup S^{\bar{a}}$*;*

2. $T^a = {}^\bullet(S^a)^\bullet$, $T^{\bar{a}} = {}^\bullet(S^{\bar{a}})^\bullet$.

We now define the synchronization of a net N via a primitive labelling π_a. An intuitive meaning of this operation is as follows. If a token of the net N reaches one place labelled by a, it may cause the continued execution of the net as if the token were in one of the places labelled by $\bar{a}$. The choice of the a-labelled place is external. It depends on the ability of the transitions adjacent to $\bar{a}$-labelled places to fire.

Notice that this intuitive meaning of place synchronization resembles a well-known synchronization via transition (e.g. see [4]), where the firing of a transition, labelled by a, causes the firing of one of the transitions labelled by $\bar{a}$.

Definition 3.3 *Let* $N = \langle S, T\rangle$ *be a net and* π_a *its primitive labelling. Synchronization of* N *w.r.t.* π_a*, called a-synchronization, is a net* $N' = \langle S', T'\rangle = (N \text{ sy } a)$ *where*

1. $S' = S \setminus S^{a\bar{a}} \ \cup\ S^a \times S^{\bar{a}}$*;*

2. $T' = \{\langle Q'(t, s^a, s^{\bar{a}}), \underline{t}, Q''(t, s^a, s^{\bar{a}})\rangle \mid t \in T, s^a \in S^a, s^{\bar{a}} \in S^{\bar{a}}\}$ *where*

$$Q'(t, s^a, s^{\bar{a}}) = {}^\bullet t \setminus S^{a\bar{a}} \cup ({}^\bullet t \cap S^a) \times \{s^{\bar{a}}\} \cup \{s^a\} \times ({}^\bullet t \cap S^{\bar{a}})$$

$$Q''(t, s^a, s^{\bar{a}}) = t^\bullet \setminus S^{a\bar{a}} \cup (t^\bullet \cap S^a) \times \{s^{\bar{a}}\} \cup \{s^a\} \times (t^\bullet \cap S^{\bar{a}})$$

Informally, a–synchronization involves the following steps:

1. The sets of a– and $\bar{a}$–labelled places (S^a and $S^{\bar{a}}$) are substituted by their Cartesian product $S^a \times S^{\bar{a}}$. We can say that each place $s^a \in S^a$ is split into $|S^{\bar{a}}|$ copies, $s^{\bar{a}}$ into $|S^a|$ copies, and after that their corresponding copies are merged.

Proceedings of the First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis (PAS '95)
0-8186-7038-X/95 $10.00 © 1995 IEEE

2. Each transition adjacent to a– and/or $\bar{a}$–places is split. There are four cases:
   (a) If transition t is adjacent to both S^a and $S^{\bar{a}}$, i.e. ${}^\bullet t^\bullet \cap S^a \neq \emptyset \neq {}^\bullet t^\bullet \cap S^{\bar{a}}$, then it is split into $|S^a| \times |S^{\bar{a}}|$ copies;
   (b) If a transition t is adjacent only to S^a, i.e. ${}^\bullet t^\bullet \cap S^a \neq \emptyset = {}^\bullet t^\bullet \cap S^{\bar{a}}$, then it is split into $|S^{\bar{a}}|$ copies;
   (c) Symmetrically, if ${}^\bullet t^\bullet \cap S^a = \emptyset \neq {}^\bullet t^\bullet \cap S^{\bar{a}}$, then the transition t is split into $|S^a|$ copies;
   (d) If ${}^\bullet t^\bullet \cap S^a = \emptyset = {}^\bullet t^\bullet \cap S^{\bar{a}}$, then t remains the same.

Note that all copies of the split transition t have the same name $\underline{t}$. Intuitively, a transition t is split if its input and/or output places are split.

Example 3.4
In Fig.1 an example of a–synchronization is depicted. The sets of a–labelled places $S^a = \{s_1, s_2\}$, and $\bar{a}$–labelled places $S^{\bar{a}} = \{s_3, s_4\}$ are replaced by $S^a \times S^{\bar{a}} = \{(s_1,s_3),(s_1,s_4),(s_2,s_3),(s_2,s_4)\}$. Places s_5, s_6, s_7, s_8 are invisible and therefore remain the same. Transitions e, f, g, h are split. For example, the transition $\langle\{s_5\}, e, \{s_1\}\rangle$ is split into two transitions $\langle\{s_5\}, e, \{(s_1,s_3)\}\rangle$ and $\langle\{s_5\}, e, \{(s_1,s_4)\}\rangle$ because s_1 is split into (s_1,s_3) and (s_1,s_4). Transitions b and c are not adjacent to labelled places and therefore are not changed.

Consider a more complex labelling function π_Δ where $\Delta = \{a,b\}$. If we apply the operations of a– and b–synchronization, then a natural question about commutativity arises.

Theorem 3.5 *Let N be a net, π_Δ its labelling function, $a,b \in \Delta$. Then*

$$((N \textbf{ sy } a) \textbf{ sy } b) = ((N \textbf{ sy } b) \textbf{ sy } a) \stackrel{def}{=} N \textbf{ sy } a \textbf{ sy } b$$

This result allows us to define a more general operation of synchronization with respect to simple labelling.

Definition 3.6 *Let N be a net and π its simple labelling with $Alph(\pi) = \{a_1, a_2, ..., a_k\}$. Then* $(N \textbf{ sy } \pi) \stackrel{def}{=} \underbrace{(N \textbf{ sy } a_1 \textbf{ sy } a_2 ... \textbf{ sy } a_k)}_{k \text{ times}}$

4 Synchronization via access points

In this section we introduce synchronization based on the notion of a place access point.

Definition 4.1 *Let $N = \langle S,T\rangle$ be a net. The set $\rho \subseteq \mathcal{P}(S)$ is called a place access point (or access point, for short) if the following conditions hold:*

1. *For each $P_1, P_2 \in \rho : P_1 \cap P_2 = \emptyset$;*
2. *For each $P \in \rho$: $s_1, s_2 \in P \Rightarrow {}^\bullet s_1 \cap {}^\bullet s_2 = s_1^\bullet \cap s_2^\bullet = \emptyset$.*

Informally, a place access point is a set of mutually disjoint subsets of places (1), with each such subset being called a *macroplace*. Two places of the same macroplace must not have common input and output transitions (2). Intuitively, if each macroplace of an access point has at least one token in one of the inner places, this state is treated as a *visible event* w.r.t. the access point.

Definition 4.2 *Let $N = \langle S,T\rangle$ be a net, ρ_1 and ρ_2 its access points.*

1. *Access points ρ_1 and ρ_2 are said to be disjoint if $||\rho_1|| \cap ||\rho_2|| = \emptyset$, i.e. they have disjoint sets of places;*
2. *Access points ρ_1 and ρ_2 are said to be comparable if $||\rho_1 \setminus \rho_2|| \cap ||\rho_2 \setminus \rho_1|| = \emptyset$, i.e. comparable access points may have common macroplaces.*

Note that disjoint access points are comparable.

Definition 4.3 *Let $N = \langle S,T\rangle$ be a net, with ρ_1 and ρ_2 its disjoint access points. Denote $S^{12} = ||\rho_1|| \cup ||\rho_2||$, $T^{12} = {}^\bullet(S^{12})^\bullet$. Then the synchronization of the net N via ρ_1 and ρ_2 is a new net $(N \textbf{ sy } \{\rho_1, \rho_2\}) \stackrel{def}{=} (N'' \textbf{ sy } \pi)$, where*

1. *$N'' = \langle S'', T''\rangle$ with*
   (a) *$S'' = S \setminus S^{12} \cup \bigcup_{P_1 \in \rho_1} P_1 \times \rho_2 \cup \bigcup_{P_2 \in \rho_2} P_2 \times \rho_1$;*
   (b) *$T'' = T \setminus T^{12} \cup \{\langle Q'(t), \underline{t}, Q''(t)\rangle \mid t \in T^{12}\}$ where*
   $Q'(t) = {}^\bullet t \setminus S^{12} \cup \bigcup_{P_1 \in \rho_1} ({}^\bullet t \cap P_1) \times \rho_2 \cup \bigcup_{P_2 \in \rho_2} ({}^\bullet t \cap P_2) \times \rho_1$
   $Q''(t) = t^\bullet \setminus S^{12} \cup \bigcup_{P_1 \in \rho_1} (t^\bullet \cap P_1) \times \rho_2 \cup \bigcup_{P_2 \in \rho_2} (t^\bullet \cap P_2) \times \rho_1$
2. *π is labelling defined over an alphabet $\Delta = \rho_1 \times \rho_2$:*

$$\pi(s) = \begin{cases} \tau, & \text{if } s \in S \setminus S^{12}; \\ \underline{(P_1, P_2)}, & \text{if } s = (s_1, P_2) \in P_1 \times \rho_2,\ P_1 \in \rho_1; \\ \overline{(P_1, P_2)}, & \text{if } s = (s_2, P_1) \in P_2 \times \rho_1,\ P_2 \in \rho_2. \end{cases}$$

Proceedings of the First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis (PAS '95)
0-8186-7038-X/95 $10.00 © 1995 IEEE

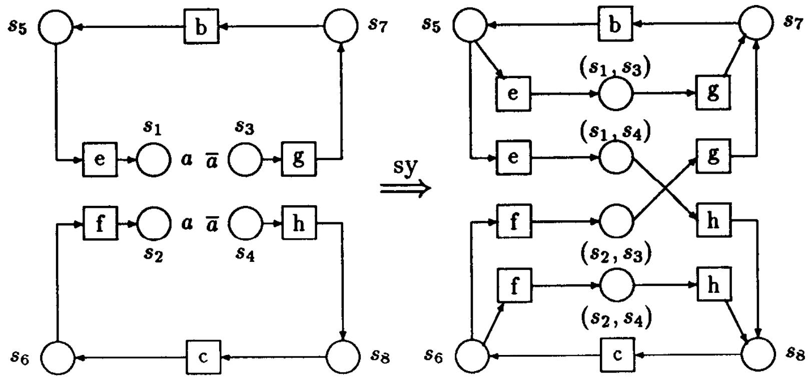

Figure 1: Example of a-synchronization

The synchronization operation is executed in two steps. In the first step, the auxiliary net N'' and its labelling π are built as follows. Each place s_1 belonging to a macroplace $P_1 \in \rho_1$ is split into $|\rho_2|$ copies, each copy corresponding to one macroplace from ρ_2. Each such copy (s_1, P_2) is labelled by the expression (P_1, P_2). Symmetrically, each place s_2 belonging to macroplace of another access point $P_2 \in \rho_2$ is split into $|\rho_1|$ copies, each copy (s_2, P_1) being labelled by the co-name $\overline{(P_1, P_2)}$. In the second step, the auxiliary net is synchronized via the new labelling π.

Example 4.4 *In Fig.2 we show an example of the synchronization operation. Initially, we have the net N and two access points ρ_1 and ρ_2 where $\rho_1 = \{\{s_1\}, \{s_2\}\}$ and $\rho_2 = \{\{s_3, s_4\}\}$. For simplicity, we will denote $x = \{s_1\}$, $y = \{s_2\}$, $z = \{s_3, s_4\}$. After the first step we obtain the net N'' and auxiliary labelling π. Since ρ_2 has only one macroplace, places s_1 and s_2 are not split*[1]*. Places s_3 and s_4 are split into two copies (s_3, x), (s_3, y) and (s_4, x), (s_4, y), respectively. After the second step, the information contained in π is used for synchronization.*

Using the above example, we can explain the intuitive meaning of the synchronization operation. The operation implements the following scheme of synchronization. If it happens that each macroplace of one access point has at least one token in one of its internal places, the synchronization leads to placing one token in each macroplace of another access point. In other words, a visible event in one access point may cause a visible event in another one.

[1] Strictly speaking, s_1 and s_2 are split into one copy, namely into (s_1, z) and (s_2, z), respectively.

Proposition 4.5 *Let N be a net and ρ_1, ρ_2 its disjoint access points. Then $(N \textbf{ sy } \{\rho_1, \rho_2\}) = (N \textbf{ sy } \{\rho_2, \rho_1\})$*

The operation of synchronization via access points substantially changes the structure of a net. However, if we still need an access point after applying the operation, then we have to redefine it trying to keep its intuitive meaning.

Definition 4.6 *Let N be a net, and ρ, ρ_1, ρ_2 its access points, such that ρ is comparable with ρ_1 and the pairs ρ_1, ρ_2 and ρ, ρ_2 are disjoint. Then we will say that the access point ρ, after application $(N \textbf{ sy } \{\rho_1, \rho_2\})$, is transferred into the new access point:*

$$\tilde{\rho} = \rho \setminus (\rho \cap \rho_1)$$
$$\cup \Big\{ P_1 \times \{P_2\} \times P_2 \times \{P_1\} \mid P_1 \in \rho \cap \rho_1, P_2 \in \rho_2 \Big\}.$$

Informally, macroplaces of ρ disjoint from ρ_1 remain the same. Each common macroplace ($P_1 \in \rho \cap \rho_1$) is transferred into $|\rho_2|$ new macroplaces, each such macroplace being formed by the Cartesian product of places from $P_1 \in \rho_1$ and $P_2 \in \rho_2$: $P_1 \times P_2$.

Proposition 4.7 *Let ρ_1 and ρ_2 be disjoint access points of the net N. Then*

$$\tilde{\rho}_1 = \tilde{\rho}_2 = \Big\{ P_1 \times \{P_2\} \times P_2 \times \{P_1\} \mid P_1 \in \rho_1, P_2 \in \rho_2 \Big\}.$$

Definition 4.8 *Let ρ_1^1, ρ_2^1, ρ_1^2, ρ_2^2 be access points of the net N. Then two pairs $\{\rho_1^1, \rho_2^1\}$ and $\{\rho_1^2, \rho_2^2\}$ are said to be comparable if ρ_1^i is comparable with ρ_2^j for some $1 \leq i, j \leq 2$, and other points are mutually disjoint.*

Proceedings of the First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis (PAS '95)
0-8186-7038-X/95 $10.00 © 1995 IEEE

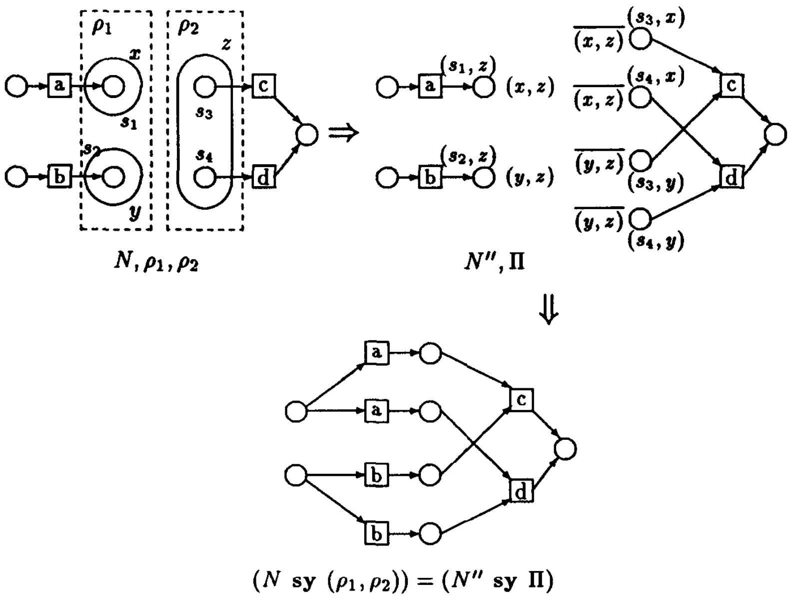

Figure 2: Example of synchronization via access point

Theorem 4.9 *Let N be a net and $\{\rho_1, \rho_2\}$, $\{\rho_3, \rho_4\}$ its comparable pair of access points. Then*

$$(N \text{ sy } \{\rho_1, \rho_2\} \text{ sy } \{\tilde{\rho}_3, \tilde{\rho}_4\}) =$$

$$= (N \text{ sy } \{\rho_3, \rho_4\} \text{ sy } \{\tilde{\rho}_1, \tilde{\rho}_2\})$$

Definition 4.10 *Let N be a net, and $\Pi = \left\{\{\rho_1^1, \rho_1^2\}, \ldots, \{\rho_n^1, \rho_n^2\}\right\}$ be a set of pairs of access points such that each pair $\{\rho_i^1, \rho_i^2\}, \{\rho_j^1, \rho_j^2\} \in \Pi$ is comparable. Then a general synchronization of the net N with respect to Π is the net:*

$$(N \text{ sy } \Pi) \stackrel{def}{=} \underbrace{(N \text{ sy } \{\rho_1^1, \rho_1^2\} \ldots \text{ sy } \{\rho_n^1, \rho_n^2\})}_{n \text{ times}}$$

Due to the Theorem 4.9 this definition is correct since the result does not depend on the order of synchronization.

5 Examples

The aim of this section is to demonstrate that the suggested synchronization technique is indeed useful and convenient. We will not introduce any problem-oriented calculus as has been done in [2]. Instead, we will give two examples of applying the technique. Note that these examples are used only for illustrative purposes. A more detailed and extensive study of these problems will be the subject for a future research.

5.1 Sequential composition

The operation of sequential composition embodies a very natural idea that one procedure can start its execution only if the other procedure has successively terminated. In order to define such an operation, one needs to formalize the notions of initial and terminal states. Usually, these states are defined as subsets of places corresponding to initial and terminal markings of the net. Sequential composition, then, comprises the cartesian products of the subsets (see [9, 4, 8]).

However, this is unsatisfactory for many applications, because there may be several initial and terminal states. Our techniques allow us to cope with this situation in a very natural way. For a net N we can define two access points ρ_h and ρ_t corresponding to initial and terminal states, called ***head*** and ***tail***. The initial (terminal) states of the net are interpreted as follows. A net is considered to be in one of the initial (resp. terminal) states if each macroplace of ρ_h (resp. ρ_t) has at least one token.

Then the ***sequential composition*** of the nets N_1 and

Proceedings of the First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis (PAS '95)
0-8186-7038-X/95 $10.00 © 1995 IEEE

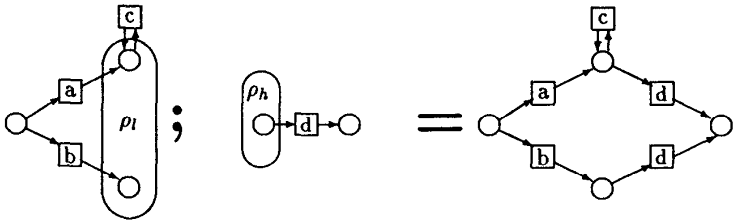

Figure 3: Example of sequential composition.

N_2 can be defined as follows:

$$(N_1; N_2) \stackrel{def}{=} (N_1 \uplus N_2 \ \mathbf{sy} \ \{\rho_{1l}, \rho_{2h}\})$$

Here $\uplus$ is a disjoint union of the two nets; ρ_{1l} is a tail access point of the net N_1; ρ_{2h} is a head access point of the net N_2 (Fig.3). Using the head and tail access points, one can relatively easily express choice (alternative composition) and iteration.

5.2 Disabling

The *disabling operation* has been introduced within LOTOS [5] and is widely used for the specification of distributed systems and communication protocols. For instance, it allows describing situations where one procedure can freely interrupt the execution of another procedure.

While the definition of this operation within the framework of an interleaving approach is easy, its definition within a net approach faces some difficulties [10, 1].

In our formalism, defining the disabling operation is quite streightforward. Suppose $N_1 = N_1^1 \uplus N_1^2 \uplus ... \uplus N_1^n$ where $n \geq 1$ and $N_1^i = \langle S_1^i, T_1^i \rangle$ is a *state-machine* (SM–) net, i.e. a net where $\forall t \in T : |{}^\bullet t| = |t^\bullet| = 1$. Define $\rho_u = \{S_1^1, S_1^2, ..., S_1^n\}$ to be an *universal* access point. Then a disabling of two nets N_1 and N_2 can be defined as follows:

$$(N_1 [\rangle N_2) \stackrel{def}{=} (N_1 \uplus N_2 \ \mathbf{sy} \ \{\rho_{1u}, \rho_{2h}\})$$

where ρ_{1u} is an universal access point of the net N_1.

In Fig.4 we show an example of disabling operation for a single SM–net.

6 Concluding remarks

In this paper we presented the basic net operations which can be used in different problem–oriented Petri net calculi. These operations generalize some other basic net operations, like multiplication of places. This work is a step towards the unification of net operations, and an attempt to distill common features and characteristics of almost all existing Petri net calculi.

The main ideas of this paper have first appeared in [1], where we designed a Petri net calculus for the specification of communication protocols, and the notion of a macroplace was introduced within a compositional context. In [2] we introduced the notion of a place access point where the access point was defined as a set of markings: $\rho = \{M_1, M_2, ..., M_k\}$. The choice between this and present definition is a matter of taste and depends on the area of application.

The problems we plan to investigate in the future include: combining place and transition synchronization operations into a common framework; relaxing some restrictions assumed in this paper; and studying problems of correctness including congruence of behavior equivalencies with respect to suggested basic operations.

Acknowledgments

The authors are grateful to Maciej Koutny for his suggestions which allowed to substantially improve the final version of the paper. The first author has also benefited from a grant from The Royal Society No. 638053.P357.

Proceedings of the First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis (PAS '95)
0-8186-7038-X/95 $10.00 © 1995 IEEE

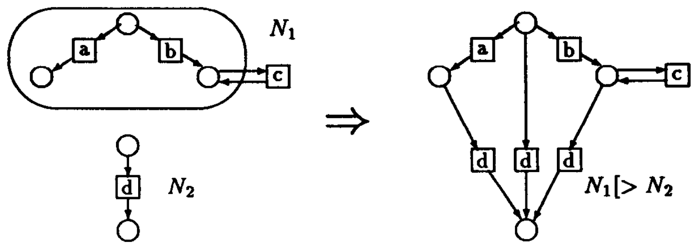

Figure 4: Example of disabling operation.

References

[1] N.A.Anisimov, An Algebra of Regular Macronets for Formal Specification of Communication Protocols. *Computers and Artificial Intelligence*, **10** (1991), 541–560.

[2] N.A.Anisimov, A.A.Kovalenko, A Synchronization of Petri Nets Via Places, Internal Report CN-93-01, *Institute for Automation and Control Processes*, Vladivostok, January 1993.

[3] J. C. M. Baeten, J. A. Bergstra, Non Interleaving Process Algebra, in: Proc. CONCUR'93, ed. E. Best, *Lecture Notes in Computer Science*, **715** (Springer-Verlag, 1993) 308–323.

[4] E.Best, R.Devillers, J.G.Hall, The Box Calculus: a New Causal Algebra with Multi-label Communication, in: *Advances in Petri Nets 1992*, ed.G.Rozenberg, *Lecture Notes in Computer Science*, **609** (Springer-Verlag, 1992) 21–69.

[5] T.Bolognessi, E.Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Network and ISDN Systems*, **14** (1987), 25–29.

[6] P.Degano, R.De Nicola, U.Montanari, A Distributional Semantics for CCS based on Condition/Event Systems. *Acta Informatica*, **26** (1988) 59–91.

[7] U.Goltz, A Class of CCS Programs Representable by Finite Petri Nets. *Lecture Notes in Computer Science*, **324** (Springer-Verlag, 1988) 339–350.

[8] R.Hopkins, J.Hall, O.Botti. A Basic-net Algebra for Program Semantics and its Application to occam, in: *Advances in Petri Nets 1992*, ed.G.Rozenberg, *Lecture Notes in Computer Science*, **609** (Springer-Verlag, 1992) 179–214.

[9] V.E.Kotov, An Algebra for Parallelism Based on Petri Nets. *Lecture Notes in Computer Science*, **64** (Springer-Verlag, 1978) 39–55.

[10] S.Marchena, G.Leon, Transformation from LOTOS specifications to Galileo Nets, in: *Formal Description Techniques. Proc. of the 1st Int. Conf. FORTE'88*, ed. K.J.Turner (North-Holland, 1988) 217–230

[11] R.Milner, A Calculus for Communication Systems. *Lecture Notes in Computer Science* **92** (Springer-Verlag, 1980)

[12] D.Taubner, Finite Representation of CCS and TCSP Programs by Automata and Petri Nets, *Lecture Notes in Computer Science*, **369** (Springer-Verlag, 1989) p.168.

Proceedings of the First Aizu International Symposium on Parallel Algorithms/Architecture Synthesis (PAS '95)
0-8186-7038-X/95 $10.00 © 1995 IEEE