Concurrency in Communicating Object Petri Nets

Rüdiger Valk
Fachbereich Informatik, Universität Hamburg
Vogt-Kölln-Str. 30, D-22527 Hamburg
email: valk@informatik.uni-hamburg.de

Abstract: Objects are studied as higher-level net tokens having an individual dynamical behaviour. In the context of Petri net research it is quite natural to also model such tokens by Petri nets. To distinguish them from the system net, they are called object nets. Object nets behave like tokens, i.e., they are lying in places and are moved by transitions. In contrast to ordinary tokens, however, they may change their state (i.e. their marking) when lying in a place or when being moved by a transition. By this approach an interesting and challenging two-level system modelling technique is introduced. Similar to the object-oriented approach, complex systems are modelled close to their real appearance in a natural way to promote clear and reliable concepts. Applications in fields like workflow, agent-oriented approaches (mobile agents and/or intelligent agents as in AI research) or open system networks are feasible. This paper gives a precise definition of the basic model together with a suitable process semantics. The focus is set more on basic concepts and their fundamental study than on high modelling capability.

1 Introduction

With the emergence of object systems and object-oriented programming also a number of papers have been published combining this modelling technique with Petri net models [Becker/Moldt 93], [Lakos 94], [Lakos 95], [Sibertin-Blanc 94]. This appears to be quite natural since both, object-oriented modelling as well as modelling by Petri nets, intend to support software development by abstraction of objects from the real world and then using the model to build a language-independent design organized around these objects. Both approaches promote better understanding of requirements, clearer designs, and more maintainable systems.

Object-oriented modelling means that software is designed as the interaction of discrete objects, incorporating both data structure and behaviour [Rumbaugh 91]. However, in most contributions, if formal techniques for describing the behaviour of objects in an object-oriented model are provided at all, these are usually equivalent to finite automata. In particular if concurrent behaviour is important, system modellers have to fall back on rather intuitive and informal methods. Here are the advantages of system modelling by Petri nets. They combine intuitive approaches with a formal treatment of systems and behavioural description. In addition they provide a deep and fundamental theory of concurrency.

From a Petri net view objects appear in the form of tokens. During the last decade tokens have been considered as more and more complex data objects. In this paper we continue our previous work [Valk 87a] by adding dynamical aspects to such token-objects. To integrate this approach into the systematics of Petri net modelling, it is quite natural to consider dynamical tokens as Petri nets themselves.

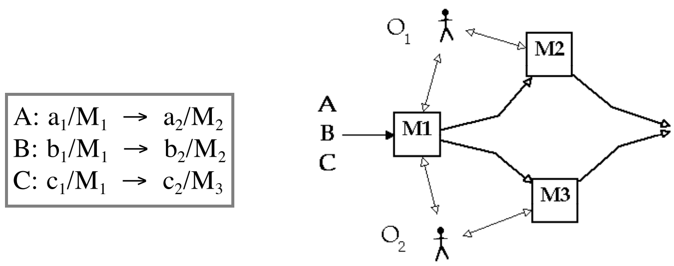

Fig. 1.1: Three machines with 3 tasks

Before giving an introduction to the formalism used and an overview on the structure of the paper we motivate the approach by some less formal examples. In the first example there are three tasks A, B and C to be processed on three machines M_1, M_2 and M_3 (fig. 1.1). There are limited resources for the machines of the following kind. Machines M_1 and M_2 are operated by an operator O_1. He can only operate one of the machines M_1, or M_2 at a given time. The same holds with operator O_2 with respect to M_1 and M_3. Machine M_1 can work, in mutual exclusion, only in a mode 1 with O_1 or in a mode 2 with O_2. Each of the tasks is divided in two subtasks, e.g. a_1 and a_2 in the case of A. The subtasks have to be executed by particular machines, as specified on the left-hand of fig. 1.1. In the case of task A the second subtask a_2 must be executed on M_2 after the execution of a_1 on M_1. We take an „object-oriented“ approach in the sense that the task is to be modelled as an object that enters machine M_1 and leaves it after execution to be then transferred to machine M_2. Attached with the object there is an „execution plan“ specifying the machines to be used and the order for doing so. Also the current „status“ of the execution is noted in the execution plan.

This situation is formalized by the Petri nets of fig. 1.2., where the two modes of machine M_1 are modelled by two different transitions M_{11} and M_{12}. Mutual exclusion is obtained by the places p_2, O_1 and O_2. Initially, all three tasks A, B and C are in the place p_1 (in the net on the right-hand side). By the „object -oriented“ approach they are not represented by an unstructured token, but by their entire execution plan, as given on the left-hand side, also as Petri nets. Note that by the marking in the nets A, B and C also their „status“ description is given.

Labels at a subtask of the form <mach11> indicate that this subtask can be executed by any machine with the same label, i.e. by M_{11} in this case. Following this convention, transition M_{11} of the „machine net“ on the right-hand side of fig. 1.2 can occur with respect to A in its input place p_1. The whole „task net“ is then moved as a token to the output place p_3, as shown in fig. 1.3. The internal token of A is also moved from s_{1A} to s_{2A} to update the current execution „status“ of the task.

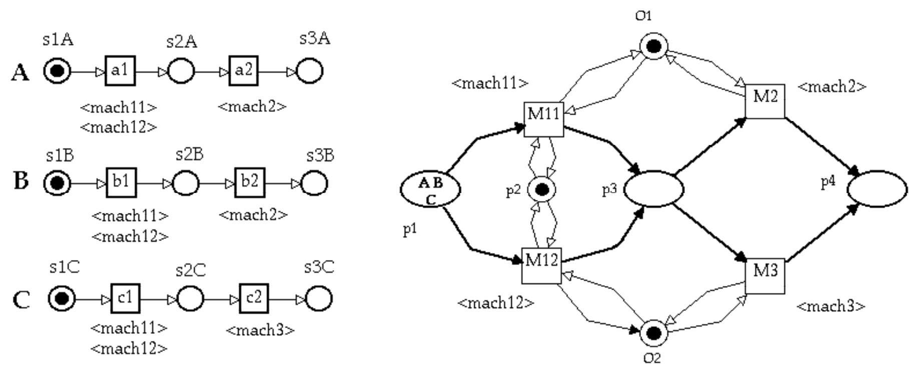

Fig. 1.2: Three machines example as object system

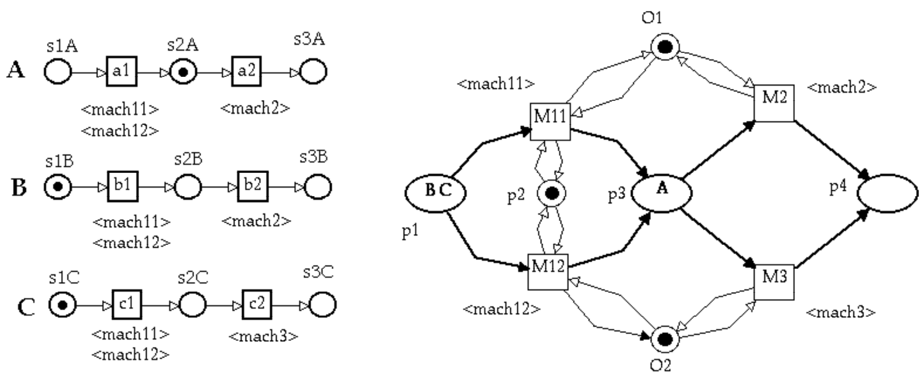

Fig. 1.3: Follower marking of the object system from fig. 1.2

To have a precise and unambiguous notation we will distinguish between a *system net* and one or several *object nets*. In the example presented before, the „machine net“ is the *system net* whereas the „task nets“ A, B and C are the *object nets*. The relation between transitions of the system net and transitions of the object net will be called the *interaction relation*. This relation is represented by labels (enclosed by sharp brackets) in the graphical representation.

In the preceding example the execution of the tasks A, B and C is performed in partial independence. This type of concurrency is restricted only by resource limitations of the system net. For instance, in the marking given in fig. 1.3 task A can be executed on machine M_2 while task B (or task C) is concurrently executable on M_{12}.

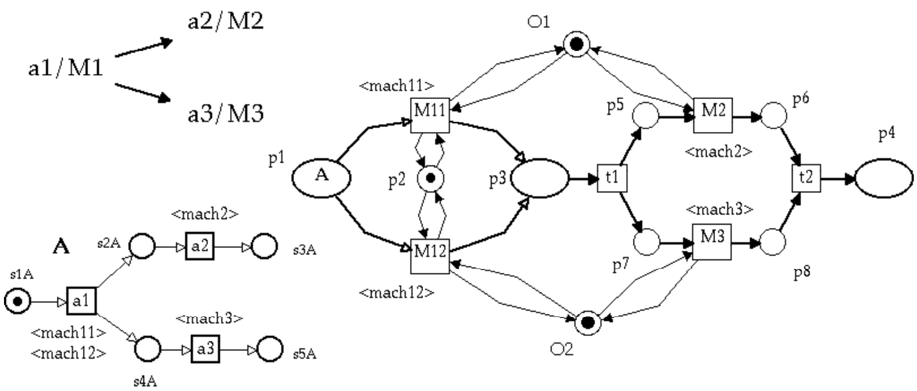

Fig. 1.4: Machine example with concurrent execution

Another type of concurrency is exemplified by the object system of fig. 1.4. In its object net, after termination of the subtask a_1 (on machine M_1), there are two successor tasks a_2 and a_3. They can be executed independently if there are suitable functional units (machines) being able to perform the execution concurrently. The system net in fig. 1.4 (on the right-hand side) offers such a suitable system architecture: after the occurrence of transition t_1 two identical descriptions of the object net (both with its current marking) are generated and placed in p_5 and p_7. With respect to the instance of the object in p_5 the subtask a_2 is executable on M_2, while concurrently the subtask a_3 of the object net instance in p_7 is executed by M_3. The „results“ of the partial executions are then „brought together“ by transition t_2, which outputs the combined and final result to the place p_4. Hence, from an intuitive point of view, M_2 and M_3 produce partial results that are independent from each other. They are „put together“ into a single task description by transition t_2. Here the precise semantics of this action in terms of Petri nets will be left open. This will be one of the results to be described in section 3. We will refer to this kind of concurrency as *intra-object concurrency*, as opposed to concurrent behaviour of two different objects.

Using standard definitions of Petri net theory the preceding examples can be encoded either as a „flat“ net by identifying corresponding transitions or as a coloured net with appropriate data type definitions in the colour sets (compare with [Moldt/Wienberg 97]). In this paper we follow a different approach: object and system nets are defined (as simple as possible) as Elementary Net Systems ("EN systems", formerly condition/event-systems), including the occurrence rule of this net type. Using the individual occurrence rules of the component nets, by combination of the individual EN-systems, *Object Systems* will be introduced, which allow to model real systems directly in the style of object oriented modelling. In doing this, we are led by the experience that has been made by the development of higher Petri nets (Pr/T-nets, coloured nets) from "lower" Petri nets (C/E-nets, P/T-nets), namely, that new features should be introduced in accordance with basic principles of concurrency theory, as formulated by C.A. Petri.

For the first time, Petri nets as dynamical objects have been considered for describing the execution of task systems in systems of functional units [Jessen/Valk 87, Valk 87a/b]. In [Valk 91] the formalism is extended in such a way that the objects are allowed to be general EN systems not necessarily restricted to (non-cyclic) causal nets.

Object-oriented modelling and programming, as appearing in current literature, is characterized by a specific object notion together with many features like generation and deletion of objects, defining classes and subclasses, inheriting attributes and many more. We do not incorporate all these features in our model as we are concentrated on such properties that can be expressed on the level of EN systems. This is done to master the complexity of the new approach. It is easier to define new models than to derive formal results. However, working on formal theories gives important hints for a suitable design of the model. Where the system net and all object nets are EN systems object systems will be called *Elementary Object Systems*.

Section 2 is concentrated on the study of object systems having only a single object net (*Unary Elementary Object Systems*). This class is introduced to investigate the behaviour of concurrent task execution. Different notions of markings are introduced. Finally a suitable formalism for the modelling of „fork/join"-concurrency structures is proposed. Also a process-oriented semantics for unary elementary object systems is given (section 3). One of the main formal results of this paper is a theorem characterizing processes of unary elementary object systems by classical processes of EN systems.

In section 4 *Simple Unary Elementary Object Systems* are studied that allow for more than a single object net. This class is restricted to system nets that are state machines. Therefore duplication of objects is not possible as well as intra-object concurrency. This restriction is not necessary. It has only been made to allow for a simpler occurrence rule and simpler semantical descriptions. On the other hand, a new feature is introduced with this model: the direct interaction of different object nets. Further complex examples using such an „inter-object communication" are presented, showing advantages of the object-oriented approach.

The main results of this contribution may be summarized as follows:

- A simple and clear notion of object Petri net is introduced such that most principles of the elementary net theory are respected.
- A formal semantics of the behaviour is given for this net class.
- It was discovered that from the different choices for the definition of markings and occurrence rules, not all of them allow a meaningful and consistent modelling of object-oriented concurrency.
- Processes of Elementary Net systems are extended to the model in a natural way. A low-level characterization of such higher-level processes is given and the equivalence is formally proven.
- Unary object systems are consistently extended to multiple objects.

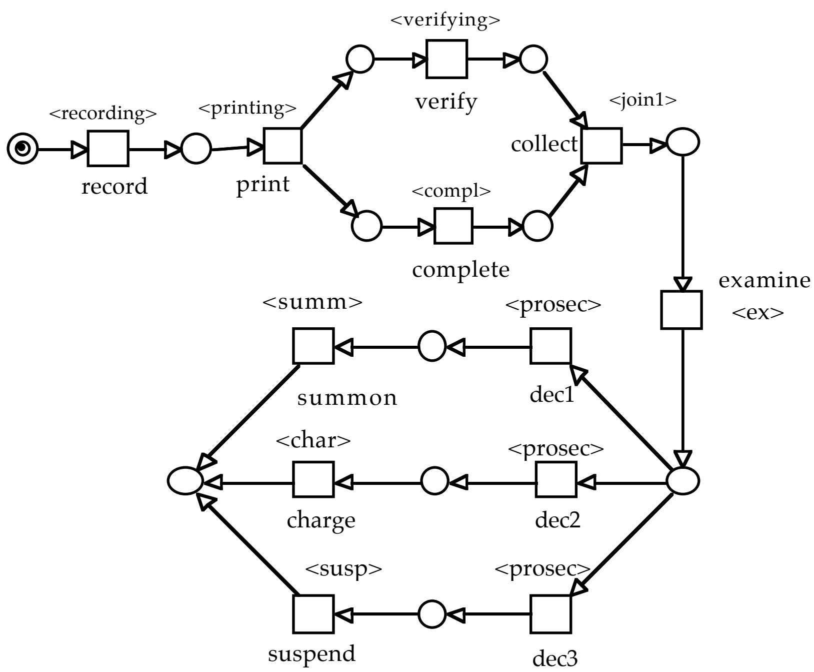

Fig. 1.5: Object net of the work flow example

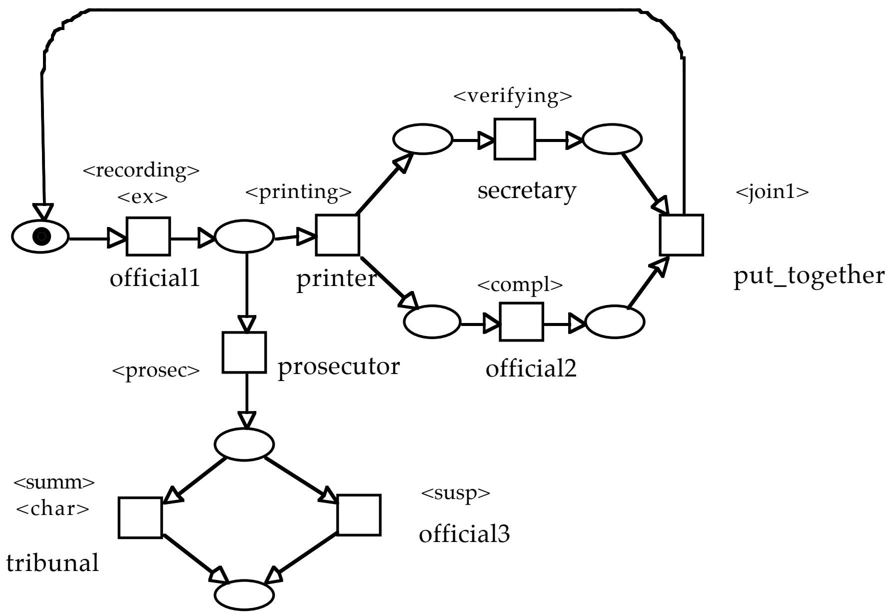

Fig. 1.6: System net of the work flow example

At the end of this introduction we now give a less artificial example of an unary elementary object system with intra-object concurrency. In the example a workflow of the Dutch Justice Department is modelled. It has been used for demonstration of modelling and analysis of workflow applications using Petri nets [Aalst 97].

The example was introduced as follows. When a criminal offence happened and the police has a suspect a record is made by an official. This is printed and sent to the secretary of the Justice Department. Extra information about the history of the suspect and some data from the local government are supplied and completed by a second official. Meanwhile the information on the official record is verified by a secretary. When these activities are completed, a prosecutor determines whether the suspect is summoned, charged or that the case is suspended.

Originally the case was modelled by a single and „flat“ net for the workflow. A slightly modified version is given as an object net in fig. 1.5. Observe that, indeed, verification and completion are concurrent subtasks. The labels in sharp brackets refer to the corresponding functional units (in fig. 1.6) executing these subtasks. For instance „printing“ is executed by a printer and „verifying“ is executed by the secretary. Official1 is executing two subtasks (record and examine) for this object net. As there are three possible outcomes of the decision of the prosecutor that are followed by different actions, the decision is modelled by three transitions dec1, dec2 and dec3.

Though being more complex the advantage of this kind of modelling by object nets lies in the direct representation of the functional units. The system net in fig. 1.6 reflects the organisational structure of the system while the object net (fig. 1.5) represents a particular workflow. Obviously there may be different workflows (object nets) for the same system of functional units (system net). The simultaneous simulation of different such executions can be used to determine bottlenecks and execution times.

2. Unary Elementary Object Systems

In this section Unary Elementary Object Systems are introduced, consisting of a *system net* SN and an *object net* ON, both being elementary net systems. These are used in their standard form as given in [Thiagarajan 87].

An *Elementary Net System* (EN system) N = (B,E,F,C) is defined by a finite set of *places* (or conditions) B, a finite set of *transitions* (or events) E, disjoint from B, a flow relation $F \subseteq (B\times E) \cup (E\times B)$, and an *initial marking* (or initial case) $C \subseteq B$. The occurrence relation for markings C_1 , C_2 and a transition t is written as $C_1[t > C_2$ or $C_1 \rightarrow_t C_2$. If t is enabled in C_1 we write $C_1[t >$ or $C_1 \rightarrow_t$. These notions are extended to words $w \in E^*$ as usual and written as $C_1[w > C_2$ (or $C_1 \rightarrow_w C_2$) and $C_1[w >$ (or $C_1 \rightarrow_w$) , respectively. N is called a *structural state machine* if each transition $t \in T$ has exactly one input place ($|{\bullet}t| = 1$) and exactly one output place ($|t{\bullet}| = 1$). N is said to be a *state machine* if it is a structural state machine and C contains exactly one token ($|C| = 1$).

$FS(N) := \{ w \in E^* \mid C [w> \}$ is the set of *firing* or *occurrence sequences* of N , and
$R(N) := \{C_1 \mid \exists\ w : C[w>C_1\}$ is the set of *reachable markings* (or cases), also called the *reachability set* of N (cf. [Rozenberg 87]). Processes of EN systems will be defined in the appendix .

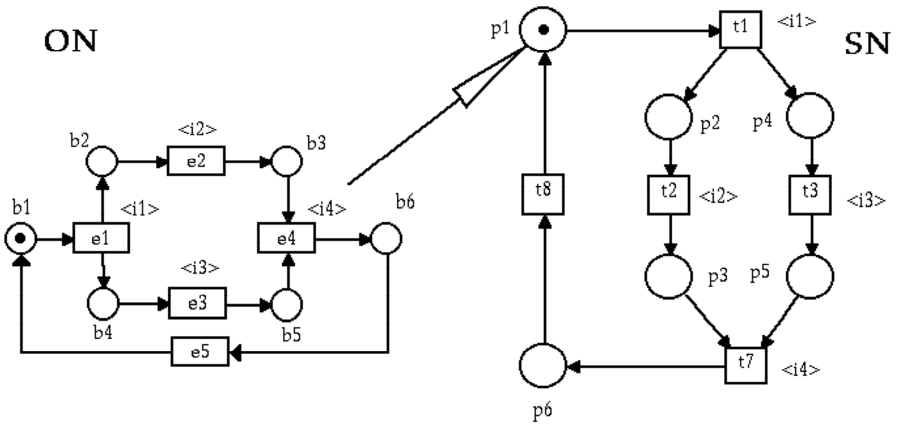

Fig. 2.1: Elementary object system "con-tasks"

Definition 2.1
An *unary elementary object system* is a tuple EOS = (SN,ON,ρ) where
SN= (P,T,W,M_o) is an EN system with $|M_o| =1$, called *system net* of EOS,
ON = (B,E,F,m_o) is an EN system, called *object net* of EOS, and
$\rho \subseteq T \times E$ is the *interaction relation*.
An elementary object system is called *simple* if its system net SN is a state machine.

Fig. 2.1 gives an example of an elementary object system with the components of an object net ON on the left-hand and a system net SN on the right-hand side. The interaction relation ρ is given by labels $<i_n>$ at t and e iff tρe ("i_n" stands for interaction number n). A similar object net is used in fig. 2.3 (i_1 is removed to illustrate autonomous transitions), but with a different system net. By this system net the "parallel" transitions e_2 and e_3 perform in a serial way. Since the system net is a state machine, the object system is simple.

Before coming to formalization we describe the intuition behind the occurrence rule to be defined afterwards. The object net ON of fig. 2.1 should be thought of lying in place p_1 of the system net SN. It is represented by a token in that place. The occurrence of transition t_1 of the system net SN should coincide with e_1 in the object net ON by the interaction i_1. The object net ON is then removed from p_1 and added to p_2 and p_4 in two copies, both of them being in the marking $\{b_2,b_4\}$. Then we observe some concurrent behaviour ending with a kind of „join“ operation by the interaction i_4 of t_7 and e_4. Furthermore, there are transitions without interaction like the so-called "autonomous" occurrences of t_8 or e_5.

In the definitions of the occurrence rule we will use the following well-known notions for a binary relation ρ. For $t \in T$ and $e \in E$ let $t\rho := \{e \in E \mid (t,e) \in \rho\}$ and $\rho e := \{t \in T \mid (t,e) \in \rho\}$. Then $t\rho = \emptyset$ means that there is no element in the interaction relation with t.

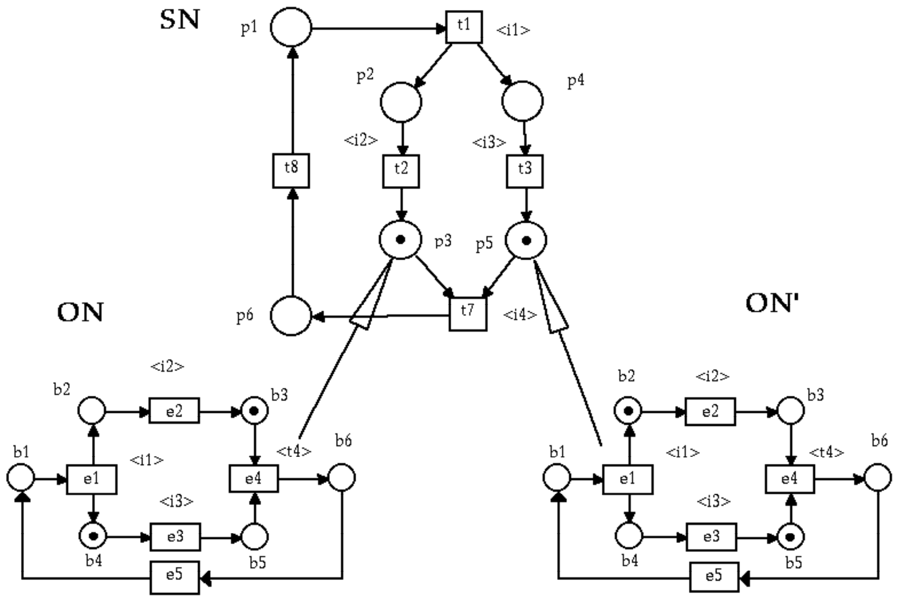

Fig. 2.2: Follower marking of fig. 2.1

Definition 2.2

A *bi-marking* of an unary elementary object system EOS = (SN,ON,ρ) is a pair (M,m) where M is a marking of the system net SN and m is a marking of the object net ON.

a) A transition $t \in T$ is activated in a bi-marking (M,m) of EOS if $t\rho = \emptyset$ and t is activated in M. Then the *follower bi-marking* (M',m') is defined by $M \rightarrow_t M'$ (w.r.t. SN) and m'=m. We write $(M,m) \rightarrow_{[t,\lambda]} (M',m')$ in this case.

b) A pair $[t,e] \in T \times E$ is *activated* in a bi-marking (M,m) of EOS if $(t,e) \in \rho$ and t and e are activated in M and m, respectively. Then the *follower bi-marking* (M',m') is defined by $M \rightarrow_t M'$ (w.r.t. SN) and $m \rightarrow_e m'$ (w.r.t. ON).

We write $(M,m) \rightarrow_{[t,e]} (M',m')$ in this case.

c) A transition $e \in E$ is *activated* in a bi-marking (M,m) of a EOS if $\rho e = \emptyset$ and e is activated in m. Then the *follower bi-marking* (M',m') is defined by $m \rightarrow_e m'$ (w.r.t. ON) and M' = M. We write $(M,m) \rightarrow_{[\lambda,e]} (M',m')$ in this case.

In transition occurrences of type b) both the system and the object participate in the same event. Such an occurrence will be called an *interaction*. By an occurrence of type c), however, the object net changes its state without moving to another place of the system net. It is therefore called *object-autonomous* or *autonomous* for short. The symmetric case in a) is called *system-autonomous* or *transport*, since the object net is transported to a different place without performing an action.

By extending this notion to occurrence sequences for the EOS of figure 2.3, for example, we obtain the following sequence: $[\lambda,e_1]$, $[t_1,\lambda]$, $[t_4,e_3]$, $[t_5,e_2]$, $[t_6,\lambda]$, $[t_7,e_4]$, $[\lambda,e_5]$. After this sequence, the initial bi-marking is reached again. We call this the *occurrence sequence semantics*. It is possible to characterize the set of all such occurrence sequences of simple EOS by some kind of intersection of the individual occurrence sequences of SN and ON. As simple object systems appear quite frequently in applications, this definition of a bi-marking and transition occurrence semantics is useful. However, the question must be asked whether it is also adequate for general EOS.

To discuss the problem consider the EOS of fig 2.1 again in a bi-marking (M,m) = ({p3,p5},{b3,b5}) that is reached after the occurrence sequence $[t_1,e_1]$, $[t_2,e_2]$, $[t_3,e_3]$. Apparently this notion of a bi-marking is not adequate since the distributed character of this state is not represented, namely it is not visible that b_3 and b_5 hold in *different* copies of the object net, as graphically visualized in fig. 2.2! In the next transition occurrence the tokens b_3 and b_5 should be used, since these tokens represent those parts of the object net processes which are the „most advanced“. It might be possible to modify the object system in such a way that the tokens b_2 and b_4 are used by a transition occurrence. This would be contraintuitive since b_2 represents a part of the process of one copy of the object net which is „less advanced“, but where the other copy was more progressive. In the same way one could argue for b_4 .

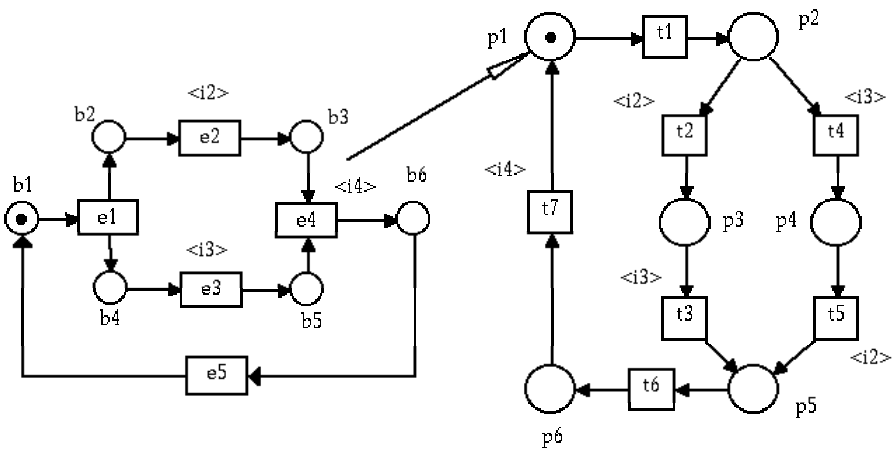

Fig. 2.3: Elementary object system "ser-task"

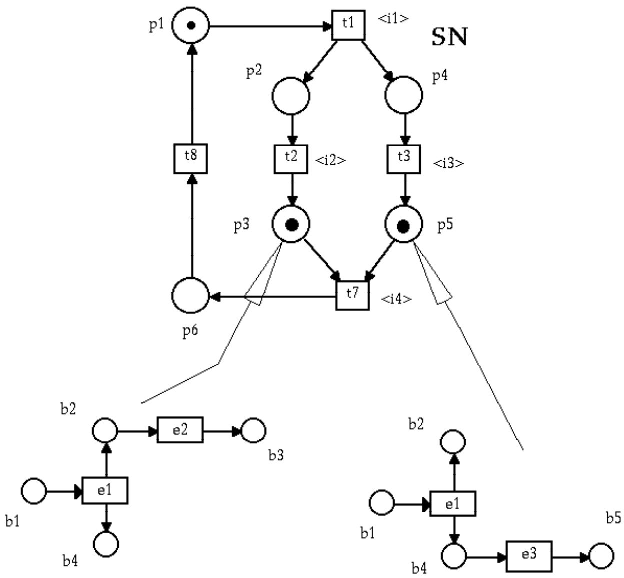

Fig. 2.4 Elementary object system "con-task" with p-marking

A solution different from bi-markings is a marking where the pairs (ON,$\{b_3,b_4\}$) and (ON,$\{b_2,b_5\}$) are assigned to p_3 and p_5, respectively (cf. fig. 2.2). As shown by a counter example in [Valk 96] also such a modelling is not adequate for non-simple unary elementary object systems. By this observation we are lead to follow a different approach for representing markings of object systems. Instead of object net markings the corresponding processes will be used.

In a bi-marking of an elementary object system, a place may be empty or contain the object net ON, being in some specific state of its execution. Such a state is now described by a process of the object net ON. Hence a marking associates a process proc $\in$ PROC(ON) to every place p $\in$ P. In order to distinguish this form of a marking from the one in the previous section, we call it a "process-marking" or in short a "p-marking".

Definition 2.3

A *process-marking* (*p-marking*) $\underline{M}$ of an elementary object system EOS = (SN,ON,ρ) is a mapping $\underline{M}$: P $\rightarrow$ PROC(ON), associating to each place of the system net SN a process proc of the object net ON (including the empty process). If in a p-marking M of an EOS $\underline{M}(p) = \emptyset$ (the empty process), we say the place is *empty*, else *occupied.* The set CM := $\{p \in P \mid \underline{M}(p) \neq \emptyset\}$ of occupied places defines a *case* or *marking* of EOS.

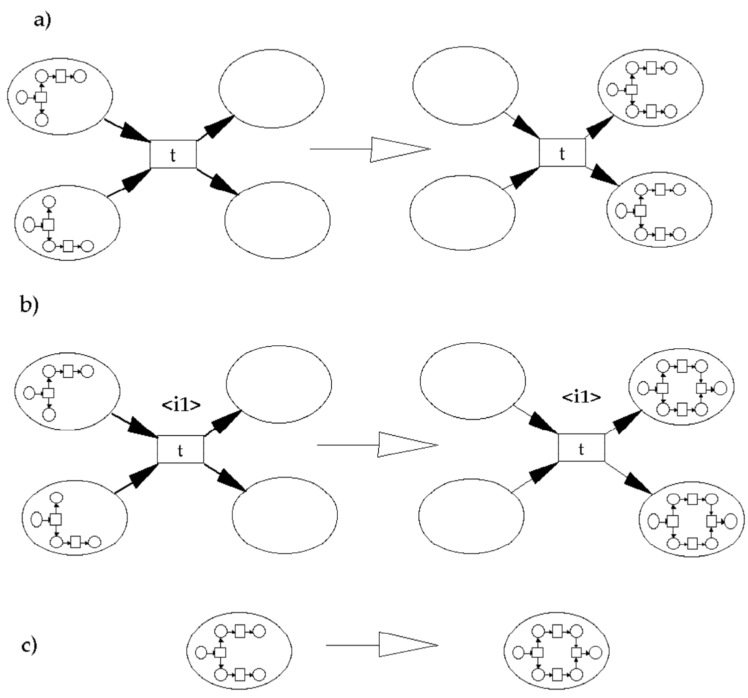

Fig. 2.5: Object system occurrence rule: a) transport, b) interaction, c) autonomous

In this definition PROC(ON) denotes the set of all processes of ON. (For definitions concerning processes see appendix.) For introducing the new occurrence rule, consider the EOS of fig. 2.1. In the initial marking the process consisting of b_1 is in the place p_1 of the system net. Now consider a follower state after the occurrence of $[t_1,e_1]$, $[t_2,e_2]$ and $[t_3,e_3]$. For the next step t_7 is enabled since all its input places are non-empty and $(t_7,e_4) \in \rho$. But in addition e_4 should be enabled in ON. The preconditions of e_4 are satisfied if *all* copies of the object net processes lying in the input places in p_3 and p_5 of t_7 are taken into consideration (see fig. 2.4). The joint information is obtained by the least upper bound „lub“ of these processes.

Definition 2.4

Let $\underline{M}$ be a p-marking of an unary elementary object system EOS, $t \in T$ a transition of the system net SN and $e \in E$ a transition of the object net ON.

To activate t, it is necessary in any of the following cases a), b) and c) that all input places $p \in \bullet t$ are occupied, the process LUB := lub($\{\underline{M}(p) \mid p \in \bullet t\}$) exists and all output places $p \in t\bullet$ are empty in $\underline{M}$.

In case a), where $t\rho=\emptyset$, transition t is activated and we write $\underline{M}\rightarrow_{[t,\lambda]}$.
In case b), where $t\rho e$, the pair [t,e] is activated (denoted $\underline{M} \rightarrow_{[t,e]}$) if e is enabled for LUB (cf. appendix).
For both cases a) and b) the follower p-marking $\underline{M}'$ is defined by $\underline{M}'(p) = \emptyset$ if $p \in \bullet t$, and for $p \in t\bullet$ we define $\underline{M}'(p) = LUB$ in case a) and $\underline{M}'(p) = LUB°e$ in case b) and $\underline{M}'(p) = \underline{M}(p)$ otherwise.
(Case a) is called a *system-autonomous* or a *transport occurrence* and is denoted by $\underline{M}\rightarrow_{[t,\lambda]} \underline{M}'$ whereas case b) is called an interaction and is denoted by $\underline{M} \rightarrow_{[t,e]}\underline{M}'$).
case c): If in some place $p \in P$ an object net transition $e \in E$ with $\rho e=\emptyset$ is activated in $proc \in \underline{M}(p)$, we write $\underline{M}\rightarrow_{[\lambda,e]}$ and define a follower marking $\underline{M}'$ by

$\underline{M}'(p) = proc°e$ and

$\underline{M}'(p') = \underline{M}(p)$ for $p' \neq p$.

(Case c) is called an *object-autonomous* or an *autonomous* occurrence and is denoted by $\underline{M}\rightarrow_{[\lambda,e]} \underline{M}'$).

In fig. 2.5 all cases a), b) and c) of the occurrence rule are represented symbolically. In this figure process inscriptions are not given. In general, however, it may depend on these inscriptions whether the process LUB exists or does not exist.

Next we extend the definition to sequences of the form $[\lambda,e_1]$ $[t_1,\lambda]$ $[t_2,e_2]$ $[t_3,e_3]$ $\in ((T\cup\{\lambda\})\times (E\cup\{\lambda\}))^*$.

Definition 2.5
For an elementary object system EOS = (SN,ON,ρ) we consider occurrence sequences $w\in Q^*$ where $Q:=\underline{T}_l\times\underline{E}_l$ and $\underline{T}_l:= T\cup\{\lambda\}$, $\underline{E}_l:= E\cup\{\lambda\}$. For such sequences and p-markings $\underline{M}$ and $\underline{M}'$ the relation $\underline{M} \rightarrow_w \underline{M}'$ is inductively defined by :
1. $\underline{M} \rightarrow_w \underline{M}$ if $w = [\lambda,\lambda]$
2. $\underline{M} \rightarrow_{wq} \underline{M}'$ if $w \in Q^*$, $q \in Q$ and $\underline{M} \rightarrow_w \underline{M}''$, $\underline{M}''\rightarrow_q \underline{M}'$ for some p-marking $\underline{M}''$.

Definition 2.6
The *initial p-marking* of EOS is defined using the initial markings M_o and m_o of SN and ON, respectively: $\underline{M}_o(p)$:= if $p\in M_o$ then m_o else $\emptyset$. (Note that in this context m_o means the initial process of EOS, as defined in the appendix).

Definition 2.7
Given an elementary object system EOS = (SN,ON,ρ), then
FS(EOS) := $\{ w \in Q^* \mid \exists\ \underline{M}: \underline{M}_o \rightarrow_w \underline{M}\}$ is the *set of occurrence sequences* of EOS, and R(EOS) := $\{\underline{M} \mid \exists\ w : \underline{M}_o \rightarrow_w \underline{M} \}$ is the set of *reachable p-markings*, also called the *reachability set* of EOS.

3. Processes of Unary Elementary Object Systems

The definition of processes of unary elementary object systems is quite obvious if autonomous occurrences are not considered. To give an example, in fig. 3.2 a process of the elementary object system "con-tasks" EOS = (SN,ON,ρ) from fig. 2.1 is constructed as follows. A process of the system net SN is extended in such a way that the places contain the object net process in the corresponding p-marking. Autonomous transitions are represented in black. Throughout this section all elementary object systems are assumed to be unary.

Concurrency of transitions is a fundamental topic of Elementary Net Systems. Transitions may occur concurrently if their input and output places are disjoint. A more interesting situation occurs if these transitions occur in markings where the object net process takes part in the system net transition occurrence. This is formally treated in lemma 3.1. and graphically represented in fig. 3.1 a) and b) for the case n=2.

Lemma 3.1
Let EOS = (SN,ON) be an unary elementary object system and „t“ a system-autonomous transition of SN and „e“ an object-autonomous transition of ON. Suppose $\bullet t = \{p_1\}$ and $|t\bullet| = n \geq 1$ and $\underline{M}_1(p_1) = proc°e$ for some p-marking $\underline{M}_1$ (i.e. p_1 contains a process where „e“ is „at the end“ (see appendix)). Then there are p-markings $\underline{M}$ and $\underline{M}'$ such that

$$\underline{M} \rightarrow_w \underline{M}' \quad \text{where } w=[\lambda,e][t,\lambda] \text{ and}$$
$$\underline{M} \rightarrow_v \underline{M}' \quad \text{where } v=[t,\lambda][\lambda,e]^n.$$

Proof: Starting from $\underline{M}_1$ p-markings $\underline{M}$ and $\underline{M}'$ are constructed as follows in a) and b):
a) Since $\underline{M}_1(p_1)$ has the form proc°e there is a predecessor marking $\underline{M}$ such that $\underline{M} \rightarrow_{[\lambda,e]} \underline{M}_1$ where $\underline{M}(p_1)$ = proc and $\underline{M}(p) = \underline{M}_1(p)$ for $p \neq p_1$
b) Since t is activated in $\underline{M}_1$ there is a follower marking $\underline{M}'$ of $\underline{M}_1$ (i.e. $\underline{M}_1 \rightarrow_{[t,\lambda]} \underline{M}'$) where proc°e is contained in all n output places of t. By a) and b) we have $\underline{M} \rightarrow_{[\lambda,e]} \underline{M}_1 \rightarrow_{[t,\lambda]} \underline{M}'$.
To prove the second part of the lemma, we observe that $[t,\lambda]$ is also activated in $\underline{M}$, since the unique input place p_1 of t contains the process proc i.e. $\underline{M} \rightarrow_{[t,\lambda]} \underline{M}_2$ for some p-marking $\underline{M}_2$. In $\underline{M}_2$ all n output places of t contain the process proc in each of which the transition e is activated. This leads to $\underline{M} \rightarrow_{[t,\lambda]} \underline{M}_2 \rightarrow_u \underline{M}'$ with $u = [\lambda,e]^n$.
q. e. d.

While object autonomous transition occurrences are not part of SN processes they should be visible in the object system process. For distinction we denote them by small and solid (black) transitions as in fig. 3.1 and 3.2. Furthermore it is straight forward to introduce an equivalent step containing both transitions as in fig. 3.1 c).

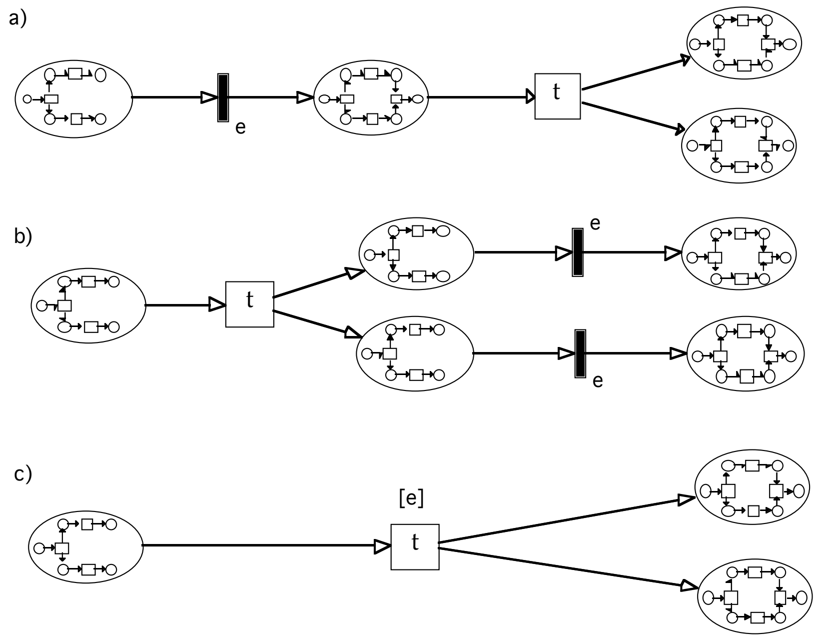

Fig. 3.1: Concurrent autonomous transitions

We will call this a reduced process equivalent.

Next we inductively define a process of an elementary object system, having a set of places $\underline{P}_\pi$ and a set of transitions $\underline{T}_\pi$, together with mappings ϕ and μ. For a place $x \in \underline{P}_\pi$ or a transition $y \in \underline{T}_\pi$ of this process $\phi(x)$ and $\phi(y)$ give the corresponding place or transition of the system net, respectively. $\mu(x)$ will be the process of the object net ON associated to a place x. Furthermore for each place $x \in \underline{P}_\pi$ the interaction relation $\rho \subseteq T \times E$ is extended to $\rho_x \subseteq T_\pi \times E_\pi$, where E_π is the set of places of the process $\mu(x)$. These mappings are given by inscriptions in the example of fig. 3.2. as explained after the following definition.

Definition 3.2
For a given firing sequence $w \in FS(EOS)$ of an unary elementary object system EOS = (SN,ON,ρ), where SN = (P,T,W,M_o), ON = (B,E,F,m_o), a *process* proc(w) = $(\underline{P}_\pi,\underline{T}_\pi,\underline{F}_\pi,\phi,\mu)$ is a structure consisting of a causal net $(\underline{P}_\pi,\underline{T}_\pi,\underline{F}_\pi)$ and mappings

$$\phi : \underline{P}_\pi \cup \underline{T}_\pi \rightarrow P \cup T \cup E \quad \text{and} \qquad \mu: \underline{P}_\pi \rightarrow PROC(ON).$$

proc(w) is defined by induction over Q* Furthermore for each object net process $proc_2 = (B_\pi,E_\pi,F_{2\pi},\phi_2) = \mu(x)$, $x \in \underline{P}_\pi$ an *extended interaction relation* $\rho_x \subseteq T_\pi \times E_\pi$ is defined.
I. If $w = \lambda$, then $\underline{P}_\pi = \{p_\pi \mid p \in M_o\}$ with $\phi(p_\pi) = p$ and $\mu(p_\pi) = m_o$ for all $p_\pi \in P$. ρ_x is empty.

(Note: markings are interpreted here as processes in the form of an initial process (see appendix).

II. Let $\underline{M}_o \rightarrow_w \underline{M} \rightarrow_{[u,v]} \underline{M}'$ and proc(w) = ($\underline{P}_\pi, \underline{T}_\pi, \underline{F}_\pi, \phi, \mu$) be the process of w. Then for $[u.v] \in Q$ we define proc(w[u,v]) = ($\underline{P}'_\pi, \underline{T}'_\pi, \underline{F}'_\pi, \phi', \mu'$) for each of the cases a), b) and c) of definition 2.4.:

a) If $[u,v] = [t,\lambda]$ and $t\rho = \emptyset$, then there is a subset $P_1 \subseteq P_\pi$ having no output transitions (i.e. $\underline{P}_1\bullet = \emptyset$) such that $\phi(\underline{P}_1) = \bullet t$. By the enabeling rule all places x in $\underline{P}_1$ contain a process $\mu(x) = proc_1$ such that their least upper bound $LUB := lub\{\mu(x) | x \in \underline{P}_1\}$ exists. To obtain ($\underline{P}_\pi', \underline{T}_\pi', \underline{F}_\pi', \phi', \mu'$) we have to do the following steps :

- a_1) Add a new set P_2 of places to $\underline{P}_\pi$ such that $\phi'(P_2) = t\bullet$, (i.e. $\underline{P}_\pi' = \underline{P}_\pi \cup P_2$).
- a_2) Add a new transition y with $\phi'(y) = t$ to $\underline{T}_\pi$ (i.e. $\underline{T}_\pi' := \underline{T}_\pi \cup \{y\}$).
- a_3) Add arcs from P_1 to y and from y to P_2 (i.e.: $\underline{F}_\pi' := \underline{F}_\pi \cup \{ (x,y) \mid x \in P_1 \} \cup \{(t',p) | p \in P_2\}$).
- a_4) Define $\phi' = \phi$ for all old places and transitions and for the new ones as defined in a_1) and a_2).
- a_5) Define $\mu'(x) = \mu(x)$ for the old places $x \in \underline{P}_\pi$ and for the new places $x_2 \in P_2$ with $\phi(x_2) \in t\bullet$ we define $\mu'(x_2) := LUB$ (i.e. the output places of t contain the same process LUB. ρ_x for $x \in P_2$ remains as in the old places.

b) If $[u,v] = [t,e]$ and $t\rho e$, then P_1 exists as in case a). The steps b_1) to b_4) are defined as a_1) to a_4), respectively.

- b_5). Define $\mu'(x) = \mu(x)$ for the old places $x \in P_\pi$ and for the new places $x_2 \in P_2$ with $\phi(x_2) \in t\bullet$ we define $\mu'(x_2) := LUB°e$ (i.e. the output places of t contain the process LUB extended by e). (y,e_2) is added to ρ_x for $x \in P_2$ and the new e_2 with $\phi_2(e_2) = e$.

c) If $[u,v] = [\lambda,e]$ and $\rho e = \emptyset$, then there is a place $x \in P_\pi$ with $x\bullet = \emptyset$ such that "e" is enabled in the process $proc_1 = \mu(x)$.

- c_1) Add a new place x_2 to P_π (i.e. $\underline{P}_\pi' = \underline{P}_\pi \cup \{x_2\}$).
- c_2) Add a new transition y with $\phi(t) = e$ to $\underline{T}_\pi$ (i.e. $\underline{T}_\pi' := \underline{T}_\pi \cup \{y\}$
- c_3) Add arcs from x to y and from y to x_2 (i.e.: $\underline{F}_\pi' := \underline{F}_\pi \cup \{ (x,y), (y,x_2)\}$).
- c_4) Define $\phi' = \phi$ for all old places and $\phi'(x_2) = \phi(x_1)$.
- c_5) Define $\mu'(x) = \mu(x)$ for the old places $x \in \underline{P}_\pi$ and for the new place $x_2 \in P_2$ we define $\mu'(x_2) := proc_1°e$ (i.e. the output places of y contain the same processes as the input places but extended by a new e_2 with $\phi_2(e_2) = e$). ρ_x is not modified.

An example of an object system process is shown in figure 3.2. In the graphical representation ϕ and μ are given as follows. Places are named x_1, x_2,... and inscriptions at such places have the form $x_i/\phi(x_i)$ (actually, due to the graphical editor: xi/ϕ(xi)). The object net process $\mu(x)$ is drawn into the ellypse of x. y_1, y_2,... denote transitions. They have inscriptions of the form

a) $[\phi(y_i)]$ if $\phi(y_i) = t_i$ is a transport, i.e. $t_i\rho = \emptyset$,
b) $[\phi(y_i),e] \in T \times E$ if $\phi(y_i) = t_i$ interacts with e, i.e. $t_i\rho e$ and
c) $[\phi(y_i)]$, $\phi(y_i) = e \in E$ if e is autonomous i.e. $\rho e = \emptyset$.

Transitions y of case c) are called *autonomous* and drawn as black rectangles.

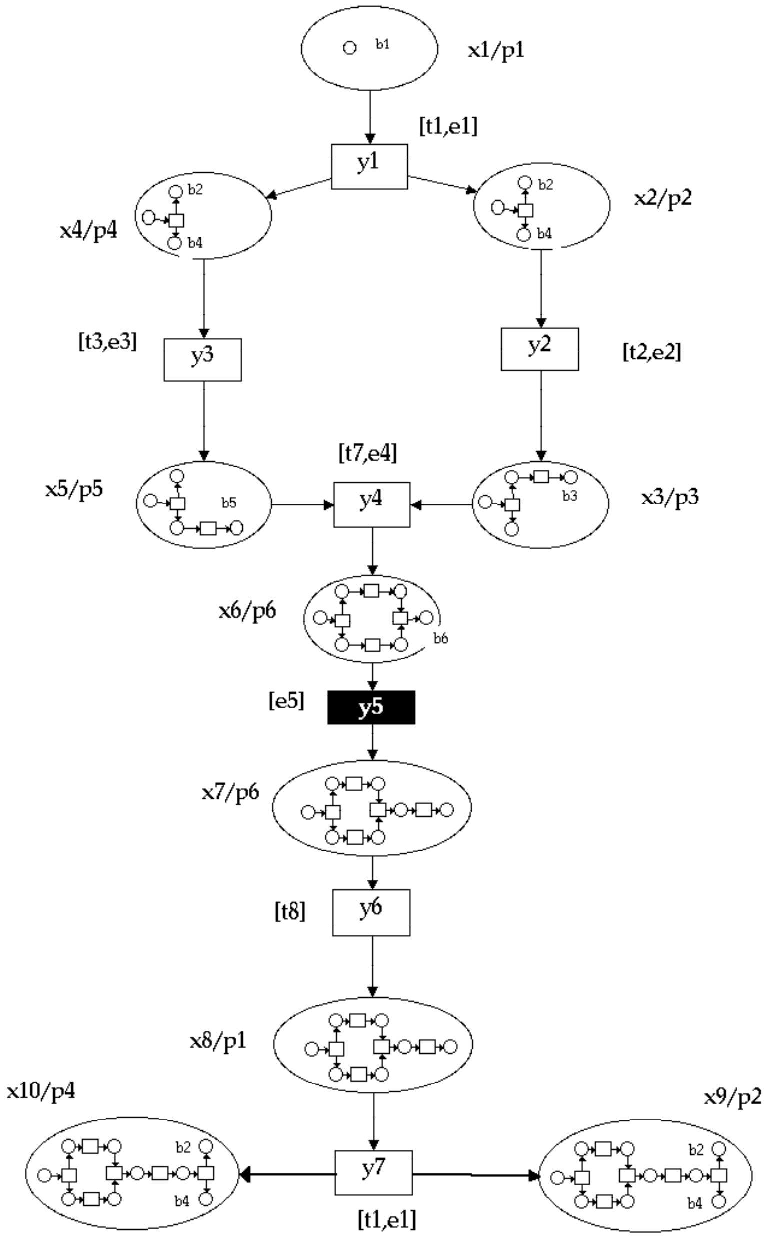

Fig. 3.2: A process of the object system "Con-Tasks" from fig. 2.1

Lemma 3.3
With the notation of def. 3.2 the following holds: for each place $x \in \underline{P}_\pi$, $proc_x = (B_\pi,E_\pi,F_{x\pi},\phi_{x\pi}) = \mu(x)$, and $e \in E_\pi$ with $\rho_\pi e \neq \emptyset$ there is some transition y with $y <_{proc(w)} x$ (i.e. y "before" x) such that $y\rho_\pi e$.

Proof: e is either introduced to $proc_x$ in $\mu(x)$ in step II b) of definition 3.2 (then $y\rho_\pi e$ for some $y \in \bullet x$) or e is created with a copy of $proc_2$ from some $x \in \bullet y$ in one of the other steps (in that case the statement holds by induction). q.e.d.

Definition 3.4
An autonomous transition y of an object system process as introduced in definition 3.2 c) has a unique input place x_1 and a unique output place x_2 with $\phi(x_1) = \phi(x_2)$. Therefore identifying x_1 and x_2 to a new place x and eliminating y gives a consistent notion of a process. For the merged place x the contained process $\mu(x)$ is defined by $\mu(x_1)$ if $x_2\bullet \neq \emptyset$ and $\mu(x_2)$ if $x_2\bullet = \emptyset$. The causal net $(\underline{P}_\pi,\underline{T}_\pi,\underline{F}_\pi,\phi,\mu)$ obtained by iterating this construction until all autonomous transitions are eliminated is said to be "*in reduced form*".

The reduced form of a process can be interpreted as process where autonomous transition occurrences are hidden.

It is a general observation in net theory that behavioural effects appear in a similar way in high level nets, e.g. Coloured Petri Nets, as in low level nets, for instance in Elementary Net Systems. It is often easier to study these effects in the low level form to profit from the gained experience for use with high level nets. This was our major motivation to search for a low level equivalent of the high level process notion of object systems. In fact, proving the theorem of this section has much influenced our efforts in finding a consistent formalisation of the object system semantics. Furthermore the theorem provides general insight into the nature of distributed computing.

A natural approach for representing processes of elementary object systems is to construct the two processes of the system and the object net side by side as co-operating processes. This is done in fig. 3.3 for the EOS "con-tasks" (fig. 2.1) and its process (fig. 3.2). This figure contains at the top a process of the object net ON from the elementary object system EOS = (SN,ON,ρ). Below a process of the system net SN is drawn. Interacting transitions t and e with tρe are connected. A formal definition follows the induction principle of def. 3.2 and is not given here in full detail.

Definition 3.5
Let EOS = (SN,ON,ρ) be an elementary object system with SN= (P,T,W,M_o) and ON = (B,E,F,m_o). Given a process proc = $(\underline{P}_\pi,\underline{T}_\pi,\underline{F}_\pi,\phi,\mu)$ of EOS (def. 3.2) having a latest place x_ω (see appendix for a definition) a *cop-process* (process in co-operating process form) of EOS is defined as a triple

$$\Theta = (proc_1,proc_2,\rho_\pi) \text{ where}$$
$$proc_1= (P_\pi,T_\pi,F_{1\pi},\phi_1) \in PROC(SN),$$

$proc_2 = (B_\pi, E_\pi, F_{2\pi}, \phi_2) \in PROC(ON)$ and
$\rho_\pi \subseteq T_\pi \times E_\pi$.

where $proc_1 = (\underline{P}_\pi, \underline{T}_\pi, \underline{F}_\pi, \phi)$ (i.e proc without μ), $proc_2 = \mu(x_\omega)$ and ρ_π as definef in def. 3.2 w.r.t. $proc_2$ and x_ω i.e. $\rho_\pi := \rho_{x_\omega}$.

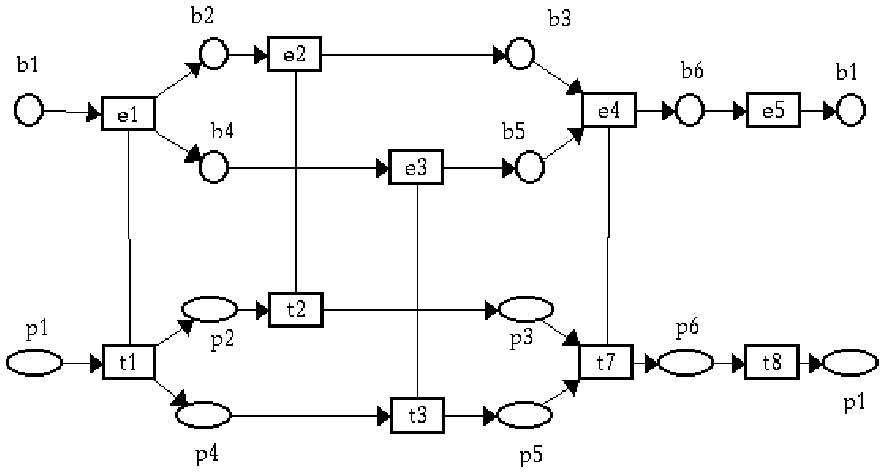

Fig. 3.3: Cooperating process representation of a subprocess from fig. 3.2

For an example consider the process of fig. 3.2 from the beginning x_1 up to the place x_7 (and delete y_6, x_8, y_7, x_9 and x_{10}). With respect to this shorter process we construct the reduced form (def. 3.4) by merging x_6 and x_7 to a new place x with $\mu(x) = \mu(x_7)$ (since $x_7\bullet = \emptyset$) and $\phi_1(x) = \phi_1(x_6) = \phi_1(x_7) = p_6$. x is a latest place x_ω. Fig. 3.3 gives the result of the construction: $proc_1$ is drawn in the lower part and $proc_2 = \mu(x_7)$ in the upper one. The definition can be extended to cover the case of the whole process of fig. 3.2 as well. Then the object net processes in the terminal cut (cf. appendix) of proc(w) should have a least upper bound LUB.

Remark

Given a cop-process $\Theta = (proc_1, proc_2, \rho_\pi)$ of an EOS as defined above, then a corresponding process of the EOS can be recovered. For $proc_1 = (P_\pi, T_\pi, F_\pi, \phi)$ and each $p \in P_\pi$ a suitable process $\mu(p)$ of ON has to be defined. This can be done by first constructing the set $T_1 := \{e \in E \mid \exists\, t \in T_\pi : t < p \wedge e\rho_\pi t\}$, where "<" is the causal order of $proc_1$. Then $\mu(p)$ is the subprocess $past_{proc1}(T_1)$ (see appendix).

Lemma 3.6

a) Given a cop-process $\Theta = (proc_1, proc_2, \rho_\pi)$ of an EOS = (SN,ON,ρ) then
$\forall\, y_1 \in T_\pi\ \forall\, e_1, e_2 \in E_\pi : y_1\rho_\pi e_1 \wedge y_1\rho_\pi e_2 \Rightarrow e_1 = e_2$ holds.

b) There is a cop-process $\Theta = (proc_1, proc_2, \rho_\pi)$ of an EOS = (SN,ON,ρ) such that
$\forall\, y_1, y_2 \in T_\pi\ \forall\, e_1 \in E_\pi : y_1\rho_\pi e_1 \wedge y_2\rho_\pi e_1 \Rightarrow y_1 = y_2$ is *not* true in general.

Proof: In the construction of ρ_π each transition $y \in T_\pi$ appears only once, whereas $e \in E_\pi$ may appear in different copies. In fig. 3.5 a cop-process of the EOS from fig.3.4 is shown together with the relation ρ_π. The cop-process fails to have property b). q.e.d.

Definition 3.7
Let be $T_{int} := \{y \in T_\pi \mid y\rho_\pi \neq \emptyset\}$ and $E_{int} := \{e \in E_\pi \mid \rho_\pi e \neq \emptyset\}$ the set of interactive transitions of $proc_1$ and $proc_2$, respectively. To simplify the following definitions and proofs from now on we exclude object autonomous transitions, i.e. we assume $E_{int}=E_\pi$.

Then (by lemma 3.6 a)) $\varphi : T_{int} \rightarrow E_\pi$ with $(\varphi(y) = e \Leftrightarrow y\rho_\pi e)$ is a mapping. φ may be non-injective (by lemma 3.6b) but is surjective, however. Hence, a *cop-process* $\Theta = (proc_1,proc_2,\rho_\pi)$ can be represented by $\Theta = (proc_1,proc_2,\varphi)$. Using this notation lemma 3.3 can be rewritten as follows.

Lemma 3.8
Given a *cop-process* $\Theta = (proc_1,proc_2,\varphi)$, then $e_1 <_{proc2} \varphi(y)$ implies
$\exists y_1 : y_1 <_{proc1} y \wedge \varphi(y_1) = e_1$.

Proof: By definition $e=\varphi(y)$ iff $y\rho_\pi e$ in the corresponding EOS-process "proc". By induction on the construction of proc transitions e_1 and e are in $\mu(p)$ for any $p \in y\bullet$. By lemma 3.3 there is a transition $y_1 <_{proc} p$ with $y_1\rho_\pi e$. By lemma 3.6 (since $y\rho_\pi e$) $y_1 \neq y$, hence $y_1 <_{proc} y <_{proc} p$, and also $y_1 <_{proc1} y <_{proc1} p$. q.e.d.

This lemma motivates a property, called *extended process morphism* property (EMP), that generalizes the notion of process morphism.

Definition 3.9
Given an elementary object system EOS = (SN,ON,ρ) and processes
$proc_1 = (P_\pi,T_\pi,F_{1\pi},\phi_1) \in PROC(SN)$, $proc_2 = (B_\pi,E_\pi,F_{2\pi},\phi_2) \in PROC(ON)$ and a mapping $\varphi: T_{int} \rightarrow E_\pi$.
φ is called *interaction true* or *true* if

a) φ is surjective,
b) $\forall\, y \in T_{int}\ \forall\, e \in E_\pi : \varphi(y) = e \Leftrightarrow \phi_1(y)\rho\phi_2(e)$
c) $\forall\, y_1,y_2 \in T_\pi : y_1 <_{proc1} y_2 \;\Rightarrow\; \varphi(y_1) \neq \varphi(y_2)$

The triple $\Theta = (proc_1,proc_2,\varphi)$ has the *extended process morphism property (EMP)* iff:

$$e_1 <\bullet_{proc2}\, e_2 \;\wedge\; y_2 \in \varphi^{-1}(e_2) \;\Rightarrow\; \exists\, y_1 <_{proc1} y_2 : \varphi(y_1) = e_1$$

($<\bullet_{proc2} \subseteq <_{proc2}$ denotes the immediate successor relation of $<_{proc2}$ restricted to transitions.)

By a) the whole object net process is reached by φ. b) relates the interaction relation of the EOS to a corresponding relation on the processes. By c) causally dependent actions are excluded to execute the same task. If φ is an injection, then $\psi: E_\pi \rightarrow T_{int}$ where $\psi:=\varphi^{-1}$ is a T-morphism (cf. appendix). There is a convincing interpretation of the extended morphism property. Consider object net transitions as tasks being executed by functional units, given here in the form of system net transitions. Then two sequential tasks e_1 and e_2 with $e_1<e_2$ cannot be executed by concurrent system net transitions (formally: $e_1<e_2$ implies $\psi(e_1) < \psi(e_2)$), as for the execution of the second task e_2 the "result" of the execution of e_1 is required. Hence concurrent object net transitions may be executed sequentially but not vice versa.

Theorem 3.10

Let be given an elementary object system EOS = (SN,ON,ρ) and a triple Θ = $(proc_1,proc_2,\varphi)$, where $proc_1 \in$ PROC(SN), having a latest place x_ω, and $proc_2 \in$ PROC(ON) are processes and $\varphi: T_{int} \rightarrow E_\pi$ is an interaction true mapping.

Then Θ is a cop-process of EOS if and only if
φ has the extended morphism property.

Proof: The necessity of the condition follows from lemma 3.8.
To prove that the condition is also sufficient, assume that $\Theta = (proc_1,proc_2,\varphi)$, where $proc_1= (P_\pi,T_\pi,F_\pi,\phi_1) \in$ PROC(SN), $proc_2 = (B_\pi,E_\pi,F_{2\pi},\phi_2) \in$ PROC(ON) are processes and $\varphi: T_\pi \rightarrow E_\pi$ is a true mapping satisfying the EMP.
First we have to find a mapping $\mu: P_\pi \rightarrow$ PROC(ON) such that proc = $(P_\pi,T_\pi,F_\pi,\phi_1,\mu)$ is an EOS-process. This is done by defining:

$$\mu(x):= past_{proc2}(\{\varphi(y_1)| \; y_1 <_{proc1} x\}) \cup init(proc_2)$$

(As defined in the appendix, $past_{proc2}(A)$ is the subprocess „generated“ by the set A.)
Next it must be shown that proc is an EOS-process, i.e. that μ is consistent with the occurrence of definition 3.2. This is done by induction on P_π w.r.t. the partial order $<_{proc}$ of proc.
a) If for $x \in P_\pi$ the condition $(\exists\ y_1 \in T_{int}: y_1 <_{proc1} x)$ does not hold then $\mu(x)$ = $init(proc_2)$ by definition.
b) (induction step) Different to case a) we may assume $(\exists\ y_1 \in T_{int}: y_1 <_{proc1} x)$, i.e. $\bullet x \neq \emptyset$, $\{y\} := \bullet x$.
<u>case</u> b_1): $y \in T_{int}$. Then $e: = \varphi(y)$ is included in $\mu(x)$ by its definition and it must be shown that the input places of y contain appropriate subnets of $proc_2$, such that their lub enables transition e. Therefore in the following we consider different subnets of $proc_2$ contained in different $\mu(x)$.
Now let be $b \in \bullet e$ an input place of e in $\mu(x)$.

<u>subcase</u> b_{11}) If there is some input transition $e_1 \in \bullet b$ then by (EMP) there is a transition y_1 with $y_1 <_{proc1} y$ and $\varphi(y_1) = e_1$. By def. 3.9 c) $\varphi(y_1) = e_1$ and $\varphi(y) = e$ are different. Hence e_1 and b also belong to an input place x_1 of y with $y_1 <_{proc1} x_1 <_{proc1} y$.

<u>subcase</u> b_{12}) If $\bullet b = \emptyset$ then $b \in init(proc_2)$ and $b \in \mu(x_1)$ for all $x_1 \in \bullet y$ by the definition of μ.

All the input places x_1 of y contain initial parts of $proc_2$. (see appendix for „initial part“). Hence the process $proc_{lub}$, defined as their lub, exists and as proved before the terminal cut of $proc_{lub}$ contains all input places of e. Therefore $proc_{lub}$ enables e (see appendix for „enables“). By similar arguments all elements from $\mu(x_1)$ are also in $\mu(x)$. This concludes the proof for case b_1).
<u>case</u> b_2): $y \notin T_{int}$. Then $\mu(x) = lub\{\mu(x_1) \mid x_1 \in \bullet y\}$ by the definition of μ. Thus the occurrence rule for EOS is also respected in this case. This concludes case b).
Finally it has to be shown that the latest place x_ω contains $proc_2$. This follows from the definition of $\mu(x_\omega)$ since by def. 3.9 a) each transition e has some $y \in \varphi^{-1}(e)$ and $y <_{proc1} x_\omega$ by the definition of the latest place. q.e.d.

In fig. 3.4 a more complex unary elementary object system is given, to illustrate the theorem by its cop-process as in fig. 3.5. The mapping φ is obviously not injective. Moreover there are system autonomous transitions (e.g. t_7). Two concurrent transitions, as t_2 and t_9 with $\varphi(t_2) = \varphi(t_9)$ may execute the same „task“ $\varphi(t_2) = \varphi(t_9) = e_3$. This redundancy can be useful in the design of reliable systems. The extended morphism property can be checked.

When simplifying the system net SN by deleting the subnet from p_5 to p_7 the corresponding process in fig. 3.5 becomes sequential and no concurrent task execution is possible any more. Then $\psi := \varphi^{-1}$ is a T-morphism. To see an example of this property consider the first occurrences of the transitions labelled e_3 and e_5. Then $e_3 < e_5$ implies $\psi(e_3) < \psi(e_5)$ which holds, since $\psi(e_3) = t_2$ and $\psi(e_5) = t_{11}$ (the labels are taken in place of the names of the transitions which are not drawn in the figure).

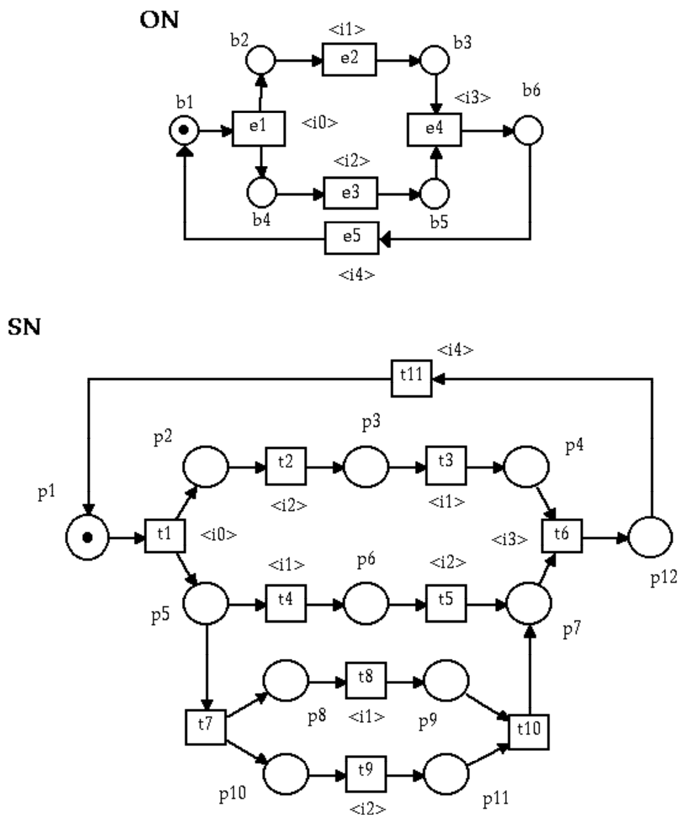

Fig. 3.4: A more complex unary EOS

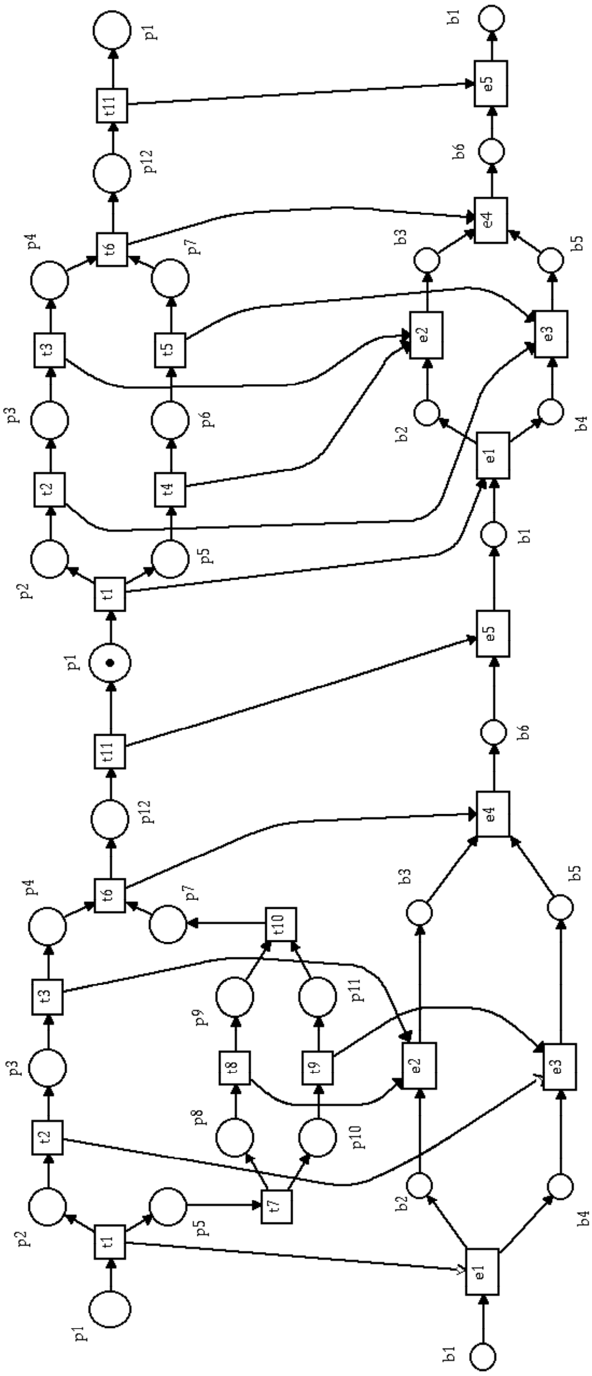

Fig. 3.5: Cop-process of the EOS from fig. 3.4

4. Elementary Object Systems

In this section unary elementary object systems are extended in such a way that different object nets are moving around in a system net and interact with both, the system net and with other object nets. As before, the model is kept as simple as possible in order to have a clear formalism.

Definition 4.1
An *elementary object system* is a tuple EOS = (SN,<u>ON</u>,Rho,type,<u>M</u>) where

- SN= (P,T,W) is a net (i.e. an EN system without initial marking), called *system net* of EOS,
- <u>ON</u> = $\{ON_1,...,ON_n\}$ ($n \geq 1$) is a finite set of EN systems, called *object nets* of EOS, denoted by $ON_i = (B_i,E_i,F_i,m_{oi})$,
- Rho = (ρ,σ) is the *interaction relation*, consisting of a system/object interaction relation $\rho \subseteq T \times \mathbf{E}$ where $\mathbf{E} := \cup\{E_i | 1 \leq i \leq n\}$ and a symmetric object/object interaction relation $\sigma \subseteq (\mathbf{E} \times \mathbf{E}) \setminus id_{\mathbf{E}}$,
- type : $W \to 2^{\{1,...,n\}} \cup \mathbb{N}$ is the *arc type* function, and
- <u>M</u> is a marking being defined in def. 4.2.

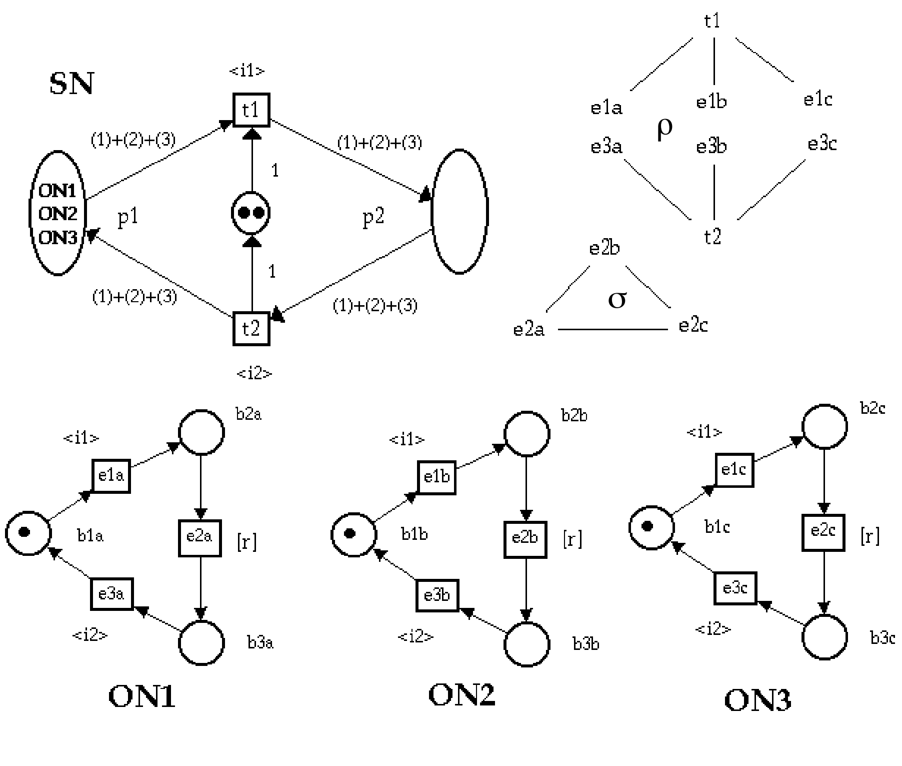

Fig. 4.1: A simple Elementary Object System with 3 object nets

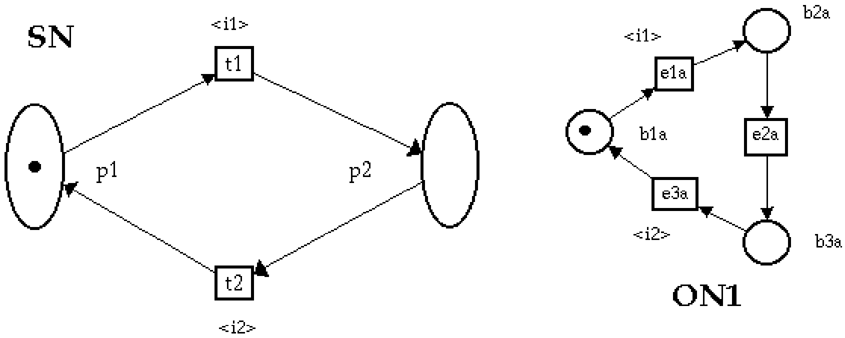

Fig. 4.2: The 1-component EOS(1) of fig. 4.1

Fig. 4.1 gives a graphical representation of an elementary object system with a system net SN and three object nets ON_i ($1 \le i \le 3$). The value of $type(p_1,t_1) = \{1,2,3\}$ is given by a corresponding arc inscription (1)+(2)+(3). Intuitively, an object net ON_i can be moved along an arc (x,y) if $i \in Type(x,y)$. Arcs of type $type(x,y) = k \in \mathbb{N}$ are labelled by $k \in \mathbb{N}$. They are used as in the case of P/T-nets. $x\rho y$ holds iff x and y are marked by the same label of the form $<i_1>$ (e.g. $t_1\rho e_{1a}$) and $x\sigma y$ is given by a label of the form [r] (e.g. $e_{2a}\sigma e_{2b}$). On the right-hand side the relation $\rho\cup\sigma$ is represented as an undirected digraph. Next, a marking will be defined as an assignment of a subset of the object nets together with a current marking to the places. It is also possible to assign a number k of tokens.

Definition 4.2
The set **Obj** := $\{(ON_i,m_i) \mid 1 \le i \le n, m_i \in R(ON_i)\}$ is the set of objects of the EOS. An *object-marking* (O-marking) is a mapping $\underline{M}$: $P \rightarrow 2^{\mathbf{Obj}} \cup \mathbb{N}$ such that $\underline{M}(p) \cap \mathbf{Obj} \neq \emptyset \Rightarrow \underline{M}(p) \cap \mathbb{N} = \emptyset$

The (initial) O-marking of the EOS in fig. 4.1 is obvious. By restriction to a particular object type from EOS we obtain a unary EOS (i-component, $1 \le i \le n$). The 0-component (zero-component) describes the part working like an ordinary P/T-net. This will be used to define simple elementary object systems.

Definition 4.3
Let EOS = (SN,$\underline{ON}$,Rho,type,$\underline{M}$) be an elementary object system as given in def. 4.1. but in some arbitrary marking $\underline{M}$.
Rho = (ρ,σ) is said to be *separated* if $i\sigma j \Rightarrow \rho i = \emptyset = \rho j$.
The *i-component* ($1 \le i \le n$) of EOS is the EN system $SN(i) = (P,T,W(i),M_{0i})$ defined by $W(i) = \{(x,y) \mid i \in type(x,y)\}$ and $M_{0i}(p) = 1$ iff $(ON_i,m_i) \in \underline{M}(p)$. The *0-component (zero-component)* is the P/T-net $SN(0) = (P,T,W(0),M_{00})$ with the arc weight function $W(0)(x,y) = k$ if $type(x,y) = k \in \mathbb{N}$ and $M_{00}(p) = k \in \mathbb{N}$ iff $k \in \underline{M}(p)$. The subnet $SN(1..n) = (P,T,W(1..n),M_{1..n})$, where $W(1..n) = \cup\{W(i) \mid 1 \le i \le n\}$ and $M_{1..n}(p) = \underline{M}(p) \cap$ **Obj** is said to be the *object-component*.
EOS is said to be a *simple elementary object system* if SN(1..n) is a structural state machine, all i-components of SN are state machines and Rho is separated.

Remark: For each $i \in \{1,...,n\}$ the *i-component* $EOS(i) := (SN(i),ON_i,\rho(i))$ is an unary EOS, where $\rho(i) := \rho \cap (T \times E_i)$.

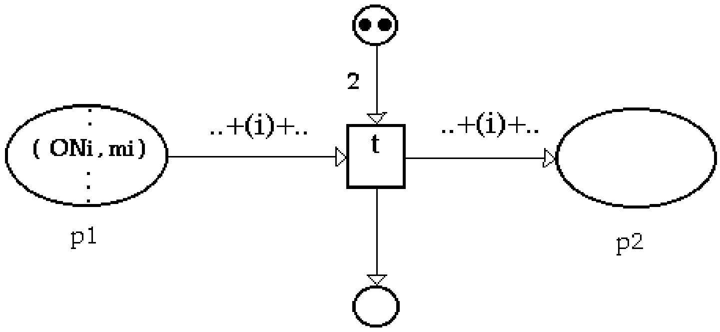

Fig. 4.3: Occurrence rule for simple EOS

The EOS from fig. 4.1 is simple since each SN(i) ($1 \leq i \leq 3$) is a state machine and Rho is separated. The latter property is easily deduced from the depicted graph of $\rho \cup \sigma$. The 1-component is a simple and unary elementary object system (see fig. 4.2). Dropping the condition that SN(1..n) is a structural state machine would lead to inconsistencies in the definition of the dynamical behaviour (def. 4.4).

By the introduction of i-components of EOS we are able to connect the models of unary EOS to general EOS. For instance, the semantical formalization of the behaviour of the more complex model of a simple elementary object system can profit from the results obtained earlier in this paper for simple unary elementary object systems. The property of separated interaction relation Rho allows to separate system/object interactions from the new concept of object/object interaction. The latter form of interaction is restricted to the case where the i-components perform autonomous transitions in the same place of the system net. Therefore in the following definition of transition occurrence of simple EOS system/object interactions are defined using case b) of def. 2.2 whereas object/object interactions are associated with case c) of this definition.

Definition 4.4

Let be EOS = (SN,$\underline{ON}$,Rho,type,$\underline{M}$) elementary object system as in def. 4.1 and $\underline{M}$: $P \rightarrow 2^{Obj} \cup \mathbb{N}$ an O-marking (def. 4.2) and $t \in T$, $e_i \in E_i$, $e_j \in E_j$, $i \neq j$.

a) Transition $t \in T$ is *activated* in $\underline{M}$ (denoted $\underline{M} \rightarrow_t$) if $t\rho = \emptyset$ and the following holds:

- a_1) t is activated in the zero-component of SN (def. 4.3) (i.e. in the P/T-net part)
- a_2) By the state machine property there is at most one type $i \in \{1,..,n\}$ such that $i \in type(p_1,t)$ and $i \in type(t,p_2)$ for some $p_1 \in {}^\bullet t$ and $p_2 \in t^\bullet$. In this case there must be some object $(ON_i,m_i) \in \underline{M}(p_1)$.(cf. fig. 4.3)

If t is activated, then t may occur ($\underline{M} \rightarrow_t \underline{M}'$) and the *follower marking* $\underline{M}'$ is defined as follows: with respect to the zero-components tokens are changed according to the

P/T-net occurrence rule. In case of a_2) (ON_i,m_i) is removed from p_1 and added to p_2 (only if $p_1 \neq p_2$).

b) A pair $[t,e] \in T\times E_i$ with $t\rho e$ is *activated* in $\underline{M}$ (denoted $\underline{M}\to_{[t,e]}$) if in addition to case a) e is also activated for ON_i in m_i. Instead of (ON_i,m_i) the changed object (ON_i,m_{i+1}) where $m_i \to_e m_{i+1}$ is added.

c) A pair $[e_i,e_j] \in E_i\times E_j$ with $e_i\sigma e_j$ is *activated* in $\underline{M}$ (denoted $\underline{M}\to_{[ei,ej]}$) if for some place $p \in P$ two objects $(ON_i,M_i) \in \underline{M}(p)$ *and* $(ON_j,m_j) \in \underline{M}(p)$ are in the *same* place p and $m_i \to_{e_i} m_{i+1}$ and $m_j \to_{e_j} m_{j+1}$. In the follower marking $\underline{M}$' the objects (ON_i,m_i) and (ON_j,m_j) in p are replaced by (ON_i,m_{i+1}) and (ON_j,m_{j+1}), respectively.

d) A transition $e \in E_i$ with $e\sigma = \sigma e = \varnothing$ is *activated* in $\underline{M}$ (denoted $\underline{M}\to_e$) if for some place $p \in P$ we have $(ON_i,m_i) \in \underline{M}(p)$ and $m_i \to_e m_{i+1}$. In the follower marking $\underline{M}$' the object (ON_i,m_i) is replaced by (OM_i,m_{i+1}).

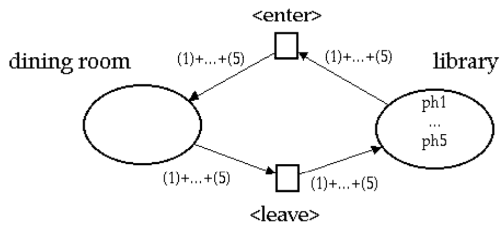

Fig. 4.4: Five Philosophers I: system net SN

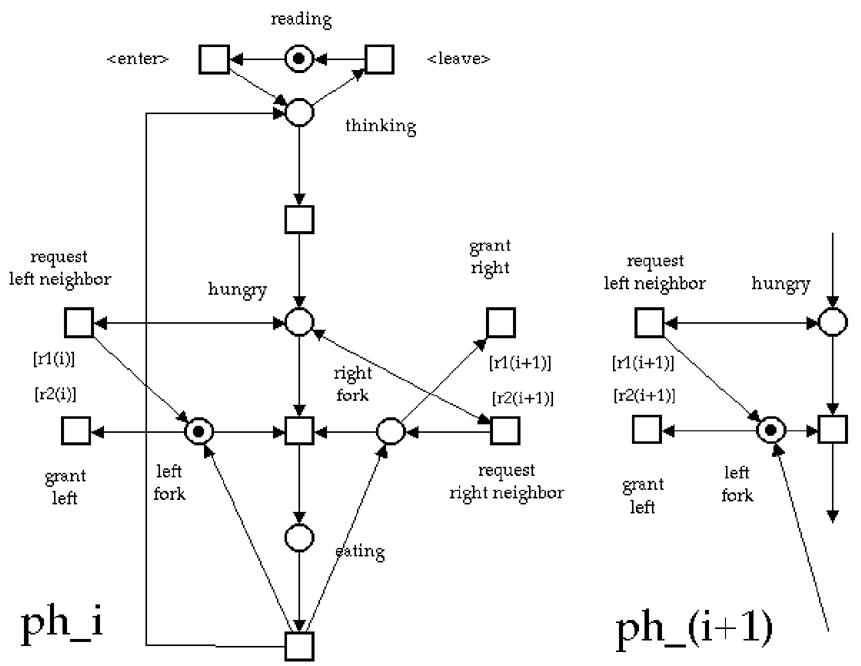

Fig. 4.5: Five Philosopers I: two object nets ph_i and ph_(i+1)

To apply the definition to a well-known example, consider the system net SN of „Five Philosophers I“ in fig. 4.4. There are five object nets $ph_1,...,ph_5$ representing the philosophers. Initially they are in a place „library“, but can „go“ by interaction <enter> into the dining room. They have their left fork in the hand when entering this room. The object nets are shown in fig. 4.5 in full detail for the instance ph_i and in part for ph_k (k=(i+1)mod5). Philosopher ph_i, for instance, being hungry can borrow the missing right fork from his right neighbour ph_k (if he is also in the dining room) by interaction of the transitions labelled [r2(k)]. All neighbouring philosophers can exchange the shared fork in the same way, under the condition that they are in the dining room. If both ph_i and ph_k are hungry at the same time, it can happen, that they permanently exchange the fork without ever managing to eat. Such „after you - after you“ effects are well known from ordinary Petri nets and can be excluded by similar methods. Solutions for this problem are out of the scope of this paper. It should be observed, however, that side conditions are used, which are not allowed by standard EN systems. By simple constructions they can be eliminated, resulting in a more complex net, however.

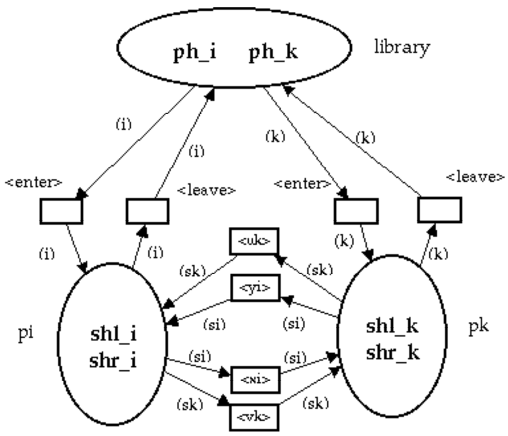

Fig. 4.6: Five philosophers II: system net SN

In a truly distributed environment the philosophers can only communicate by sending messages. This is assumed for the system of „Five Philosophers II“. An extract of the system net is given in fig. 4.6. Each philosopher ph_i can enter his own place p_i by an arc of type (i). In p_i he finds a „fork shuttle right“ shr_i, that can be used to send a request to his right neighbour ph_k by the interaction [ai] (see fig. 4.7).

The shuttle then moves to p_k using interaction $<x_i>$ to take the fork of ph_i using interaction [ek], provided philosopher ph_k is now at his place and the fork is free. Then it goes back, delivering the fork to ph_i by [ci]. The type of this object net is (si). In a symmetrical way ph_k uses shuttle shl_k („fork shuttle left“) to obtain the fork back.

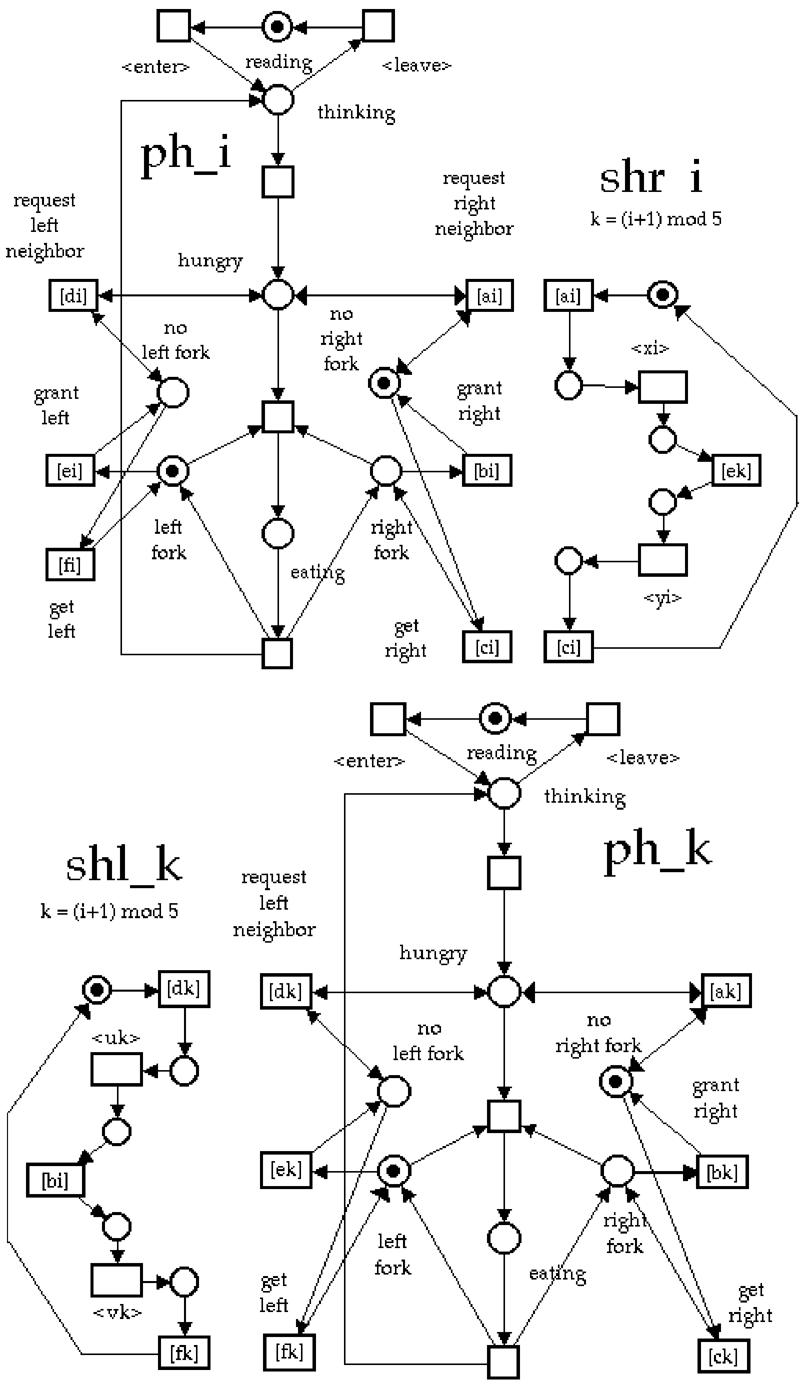

Fig. 4.7: Five Philosophers II: four object nets ph_i, shr_i, ph_k and shl_k
(k = (i+1) mod 5)

Since the partners for communication are fixed in this example by merging communicating transitions, an ordinary net (see fig. 4.8) can be constructed, representing the behaviour of shuttle exchange. This net can be seen as a communication protocol for distributed mutual exclusion, being similar to the method of [Valk 87c] and [Kindler 95].

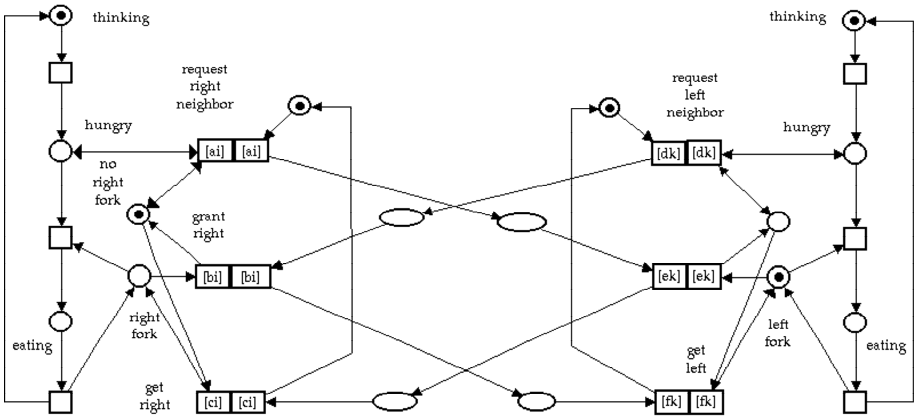

Fig. 4.8: EN system equivalent of fig. 4.7

Many different settings of the distributed philosophers problem could be realized, as well. For instance, a fork shuttle could move around and distribute forks to arbitrary participants. Also, different approaches for handling forks on leave of the dining room could be realized (e.g.: a philosopher leaves with „his“ left fork, as he came in, or he leaves without forks granting the resource to a neighbour.) Such variants of specifications are out of the scope of this paper.

The semantical description techniques discussed in sections 2 and 3 can be extended to the model of general EOS. In particular, the description of processes by (an extended version of) co-operating processes has been applied to this case and has been proved to be very useful [Valk 96]. Further research is necessary, in order to well understand the behaviour of non-simple (general) Elementary Object Systems.

5. Conclusion

An intuitive notion of object system is introduced and then formalized. *Unary object systems* are restricted to contain only a single object net, but allow for „intra-concurrency“ to model concurrent task execution. Using net processes a suitable definition of marking was found. Processes of unary object systems have been defined and were represented as *cop-processes*. This representation was characterized by the necessary and sufficient *extended morphism property*.

In the second part *simple elementary object systems* have been considered, where intra-concurrency is excluded, but concurrent behaviour and interaction of different object nets is possible. Such object systems are used to model two instances of distributed Five Philosophers, showing the usefulness of the approach for a simple and direct way to model in the object paradigm on the level of classical Petri nets.

6. Appendix: Processes of EN Systems

The non-sequential behaviour of EN systems is given by causal nets (occurrence nets (cf [Rozenberg 87])).

A process of an EN system N = (B,E,F,C) is defined by a node-labelled causal net N_π $=(B_\pi,E_\pi,F_\pi,\phi)$ such that $\phi : B_\pi \cup E_\pi \rightarrow B \cup E$ satisfies

a) $\phi(B_\pi) \subseteq B$ and $\phi(E_\pi) \subseteq E$ and $\forall\, t \in E_\pi$: $[\phi(\bullet t) = \bullet\phi(t)$ and $\phi(t\bullet) = \phi(t)\bullet]$
b) ϕ is injective on every B_π-cut of N_π
c) $\forall\, b \in B_\pi : \bullet b = \emptyset \Leftrightarrow \phi(b) \in C$

The initial process of N_π $init(N_\pi)=(B_C,E_C,F_C,\phi_C)$ consists just of the initial case C, i.e. $B_C= C$, $E_C = F_C = \emptyset$ and $\phi_C(b) = b$. The set of places $term(N_\pi):= \{b\in B_\pi |\ b\bullet=\emptyset\}$ is called a *terminal cut*. (It is assumed that all transitions $e\in E_\pi$ have an output place: $e\bullet\neq\emptyset$). Only finite processes are considered in this paper.

We use PROC(N) to denote all processes of N together with the "empty process" $\emptyset$. By $<_{N\pi} := (F_\pi)^+$ we denote the partial order "before". A place b_ω is called *latest place*, if all other places are before b_ω, i.e.: $\forall\, b \in B_\pi \backslash b_\omega : b <_{N\pi} b_\omega$. Given N_π and a subset $A\subseteq E_\pi$ of process transitions, then $past_{N\pi}(A)$ is the subnet generated by A. The process $past_{N\pi}(A) = (B'_\pi,E'_\pi,F'_\pi,\phi')$ is defined by all transitions "before or in A", i.e. $E'_\pi :=$ $\{t\in E_\pi | t\in A \ \vee\ \exists t_1\in A\colon t<_{N\pi}t_1\}$ and all input or output places of E'_π, i.e.: $B'_\pi:=$ $\{b\in B_\pi | \exists b\in E'_\pi : b \in \bullet t \cup t\bullet \}$. ϕ' is the restriction of ϕ to $B'_\pi \cup E'_\pi$. Note that $A=\emptyset$ implies $B'_\pi = E'_\pi = \emptyset$.

Given two processes $N_{\pi1}$ and $N_{\pi2}$, then a *T-morphism* from $N_{\pi1}$ to $N_{\pi2}$ is a mapping $\alpha : E_{\pi1} \rightarrow E_{\pi2}$ such that $\forall\, x,y \in E_{\pi1}$: $x <_{N\pi1} y \Rightarrow \alpha(x) <_{N\pi2} \alpha(y)$. Every firing sequence $w= e_1...e_n \in FS(N)$ uniquely deter-minates a process $proc(w) = (B_\pi,E_\pi,F_\pi,\phi)$ such that $\phi(e_i) <_{proc(w)} \phi(e_j) \Rightarrow i<j$. On the set PROC(N) of all processes there is a partial order $\leq_\pi$ "initial part": $proc_1 \leq_\pi proc_2$ if $\exists\ w_1,w_2 \in FS(N) : proc_1 = proc(w_1)$ and $proc_2 = proc(w_2)$ and $\exists\ v \in E^* : w_1v = w_2$. If in this definition $v = e \in E$, then we say that e is enabled by $proc_1$ (denoted $proc_1 \rightarrow_e$) and $proc_1{}^\circ e := proc_2$ (i.e. $proc_1{}^\circ e$ is the prolongation of $proc_1$ by a transition e_2 with $\phi(e_2) = e$ and the output places of e_2). If for two processes $proc_i$ ($i \in \{1,2\}$) there is a process $proc_3$ such that $proc_i \leq_\pi proc_3$ $(1\leq i\leq 2)$, then there is a least upper bound of $proc_1$ and $proc_2$, which we denote $lub(proc_1,proc_2)$. This definition is extended to finite subsets $Q \subseteq PROC(N)$ and denoted by lub(Q).

8. References

[Aalst 97] W. v. d. Aalst: private communication, 1997

[Becker/Moldt 93] U. Becker, D. Moldt: Object-Oriented Concepts for Coloured Petri Nets. in Proc. IEEE Int. Conf. on Systems, Man and Cybernetics, 1993, pp. 279-286

[BRR 87] W. Brauer,W. Reisig, G. Rozenberg (eds) : Petri Nets: Central Models and their Properties, Lecture Notes in Computer Science No 254, 255, Springer, Berlin, 1987.

[Kindler 95] E. Kindler,R. Walter: Message Passing Mutex, In J. Desel (Ed.): Structures in Concurrency Theory, Proceedings, Workshops in Computing, Springer-Verlag 1995..

[Jessen/Valk 87] E. Jessen, R. Valk : Rechensysteme, Springer, Berlin, 1987

[Lakos 94] C.A. Lakos: Object Petri Nets, Technical Report TR94-3, Computer Science Department, University of Tasmania,1994

[Lakos 95] C.A. Lakos: From Coloured Petri Nets to Object Petri Nets, in G. De Michelis and M. Diaz (Eds): Application and Theory of Petri Nets 1995, LNCS No. 935, Springer Berlin 1995, pp 278-297

[Moldt/Wienberg 97] D. Moldt, F. Wienberg: Multi-Agent-Systems Based on Coloured Petri Nets, in P. Azema, G. Balbo (Eds): Application and Theory of Petri Nets 1997, LNCS Vol. 1248, Springer Berlin 1997, pp 82-101

[Rozenberg 87] G. Rozenberg: Behaviour of Elementary Net Systems, in [BRR 87], part I, pp 60-94

[Rumbaugh 91] J. Rumbaugh et al.: Object-Oriented Modeling and Design, Prentice-Hall, London, 1991

[Sibertin-Blanc 94] C. Sibertin-Blanc: Cooperative Nets, in Valette, R (Ed): Application and Theory of Petri Nets 1994, LNCS Vol. 815, Springer Berlin 1994, pp 471-490

[Thiagarajan 87] P.S. Thiagarajan: Elementary Net Systems, in [BRR 87], part I, pp 26-59

[Valk 87a] R. Valk: Nets in Computer Organisation, in [BRR 87], part II, pp 218-233.

[Valk 87b] R. Valk: Modeling of Task Flow in Systems of Functional Units, report FBI-HH-B-124/87, Univ. Hamburg, 1987.

[Valk 87c] R. Valk: On Theory and Practice: an Exercise in Fairness. In: Petri Net Newsletter No. 26, pp. 4-11. Bonn, Germany: Gesellschaft für Informatik (GI), Special Interest Group on Petri Nets and Related System Models, April 1987.

[Valk 91] R. Valk: Modeling Concurrency by Task/Flow EN Systems, Proceedings 3rd Workshop on Concurrency and Compositionality, GMD-Studien Nr. 19, Gesellschaft f. Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1991

[Valk 95] R. Valk: Petri Nets as Dynamical Objects, Proc. Workshop on Object-Oriented Programming and Models of Concurrency, Torino, June, 1995

[Valk 96] R. Valk: On Processes of Object Petri Nets, Report 185/96, Fachb. Informatik, Univ. Hamburg, Hamburg, 1996