Fachbereich Informatik der Universität Hamburg
Vogt-Kölln-Str. 30, D-22527 Hamburg / Germany
University of Hamburg - Computer Science Department

Bericht Nr. 185/96 • Report No. 185/96

On Processes of Object Petri Nets

Rüdiger Valk

FBI-HH-B-185/96

Juni 1996

In die Reihe der Berichte des Fachbereichs Informatik aufgenommen durch

Accepted for Publication in the Report Series of the Department of Computer Science by

Prof. Dr. Christopher Habel and Prof. Dr. Joachim W. Schmidt

Kurzfassung:

Es werden Petrinetze untersucht, deren Marken strukturierte Objekte mit eigenständigem, dynamischem Verhalten sind. Vom Standpunkt der Petrinetztheorie ist es naheliegend, solche Marken ihrerseits als Petrinetze aufzufassen, wodurch sich ein anspruchsvolles und interessantes zweistufiges Systemmodell ergibt. Um sie besser von dem zugrundeliegenden Systemnetz zu unterscheiden, werden solche Objekte Objektnetze genannt. Objektnetze verhalten sich einerseits wie gewöhnliche Marken, d.h. sie liegen auf Stellen und werden durch das Schalten von Transitionen bewegt, andererseits können sie ihren Zustand (d.h. ihre Markierung) verändern. Dies kann sich ereignen, wenn sie in einer Stelle ruhen oder durch eine Transition verlagert werden.

Ähnlich wie bei der objektorientierten Programmierung können mit diesem Ansatz komplexe Systeme relativ natürlich und realitätsnah abgebildet werden, um klarere und zuverlässige Konzepte zu unterstützen. Anwendungen im Bereich der Modellierung von Geschäftsprozessen (workflow), agentenorientierten Programmierung (im Sinne der KI) oder in offenen Netzen bieten sich an.

In diesem Bericht werden zum ersten Mal das Grundmodell exakt definiert, eine präzise Prozeßsemantik in verschiedenen äquivalenten Darstellungen angegeben und elementare Erscheinungen von verteiltem Prozeßverhalten untersucht. Als Beispiel für eine spätere Erweiterung auf n-stufige Modelle wird ein 3-stufiges Objektsystem diskutiert.

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 model such tokens also 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 a pretentious and challenging two-level system modelling technique is introduced. Similar to object-oriented programming, complex systems are modelled close to their real appearence in a natural way to promote clear and reliable concepts. Applications in fields like workflow, agent-oriented programming (as in AI research) or open systems networks are feasible. This report gives, for the first time, a precise definition of the basic model together with a suitable process semantics. Different but equivalent notions of such processes are studied as well as some of their elementary properties. To outline further research on n-level models, a 3-level object system is discussed.

Keywords:

higher-level net model, causality/partial order theory of concurrency, system design using nets, object nets, processes, elementary net systems, interleaving and partial order process semantics

Contents

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. 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]. From a Petri net view such 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 tokens. To integrate the systematics of Petri modelling, it is quite natural to consider dynamical tokens as Petri nets themselves.

In this paper we study dynamical objects in Petri nets from the very basic level of "Elementary Net Systems" (formerly condition/event-systems). 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 invented 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]. A task system is a finite set of tasks, partially ordered by a precedence relation. It can be represented by an Elementary Net System (EN system) [Thiagarajan 87] in the form of a causal net (occurrence net). 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.

In this paper we first introduce elementary object systems composed of a *system net* with one token. The token itself is interpreted as a Petri net, called *object net*. Both nets are EN systems. This simple type of object net allows studying very elementary features of object systems. New copies of the object net may be created and distributed all over the system net. It is challenging to study here fork/join-structures, where copies sent off will join after some remote computation. To properly model such effects a particular process semantics is required going far beyond the standard interleaving semantics. Object-elementary object systems start with multiple objects. While these objects are still EN systems, the system net is modelled as a coloured Petri net.

The focus of the paper is on the study of fundamental properties. Different approaches

for occurrence rules are discussed and a notion of process semantics is introduced. High level processes are transformed into pairs of cooperating EN system processes. The formal proof of the equivalence of the two representations strongly supports the conjecture that the chosen semantics is consistent with standard net theory. Furthermore the new representation is much smaller and easier to verify. The advances with respect to [Valk 91] consist in the inclusion of autonomous transition occurrences, a more elaborated formalism and better examples.

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

- A simple and clear notion of *Object Petri Net* is introduced such that most principles of elementary net theory are preserved.
- A formal semantics of the behaviour is given for this net class.
- It was found that from the different choices for the definition of a *marking* and an *occurrence rule*, 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 proved.
- By an example, further research on n-level object systems is motivated.

Note that the formalism presented here does not mimic principles of object-oriented programming (as given in [Rumbaugh 91], but is a starting point for fundamental research on the properties of objects moving partially independent in a system.

2. Object Systems

Object-oriented programming is known to provide modelling techniques where systems and their objects are closer to the corresponding real system including real objects. Contrary to traditional programming the objects in the model individually change their states. This may be considered in opposition to the description and semantics of the basic system. As a consequence the models are simpler and more natural to interpret.

In Petri net theory we are used to design systems. However, rather little is known of "objects" in this sense. Usually objects, such as variables or records are modelled by individual tokens in high-level Petri nets. They are, however, passive data structures without an individual behaviour.

A natural further step is to consider individual tokens as descriptions of nets in a particular marking. Hence, Petri nets are used in different roles and it is important to carefully distinguish them. In the following a net modelling an object will be called an *object net.* Such an object net is seen as a token in a so-called *system net,* which is designed to model the underlying basic system.

To give an intuitive example we recall the well-known example of fire extinction given by C.A. Petri. As shown in fig.1, a number of (in this instance seven) firemen connect a water resource to the fire by a chain of buckets. The first fireman fills a bucket, makes some steps in the direction of the fire. Then he exchanges with the second fireman the full bucket against an empty one. The net on the top of fig. 2 represents the routes of firemen. The first cycle, for instance, contains the route-sections labelled <x1>,<x2> and <a1>, <a2>. Particular places between these sections mark the water source and the meeting point "a" with the next fireman. Similar routes are contained for the other firemen (in this instance four). Fireman 4 has access to the burning house in the place "fire".

We consider this net as the *system net* of the object system. It contains four *object nets* (the firemen) at the marked places. The object nets N_{f1} to N_{f4} representing the firemen are drawn separately, but should be thought of being in the corresponding places of the system net. Each fireman f_i may be in two different states, namely the states q_i and p_i. If fireman 1, for example, is in his state q_1 (q_1 is marked), then he is ready to walk to the left (enabling the transition labelled <a1,a2>) or to fill the bucket (enabeling the transition labelled <F>) .

The behaviour of these nets can be studied independently, but it is in reality not independent from the system. To give an example, when fireman f_1 makes a first step by the occurrence of transition "B", this is coincident with its moving by transition "A" in the system net. Hence transitions "A" and "B" should occur together. This connection is represented by a relation ρ. The holding of AρB is given in the graphical net representation by a common label, namely <x1> in this case. The interactions <x1>,<x2> and <a1>, <a2> fix the path on which fireman 1 is walking around. In a similar way all four fire-

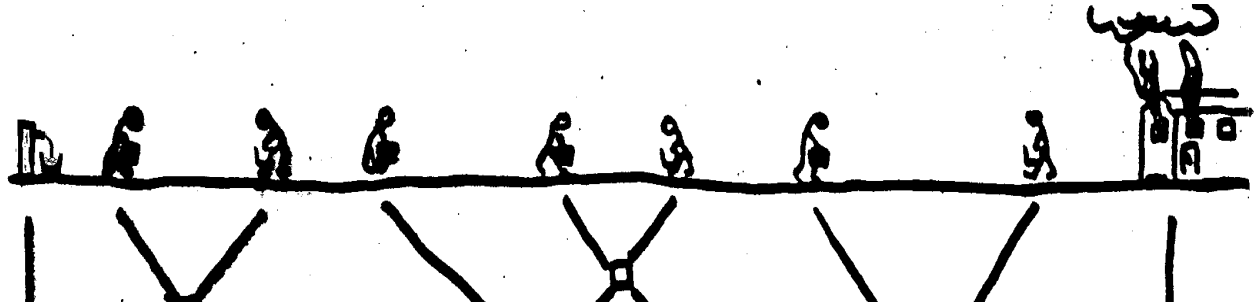


Fig. 1: Petri's fire extinction

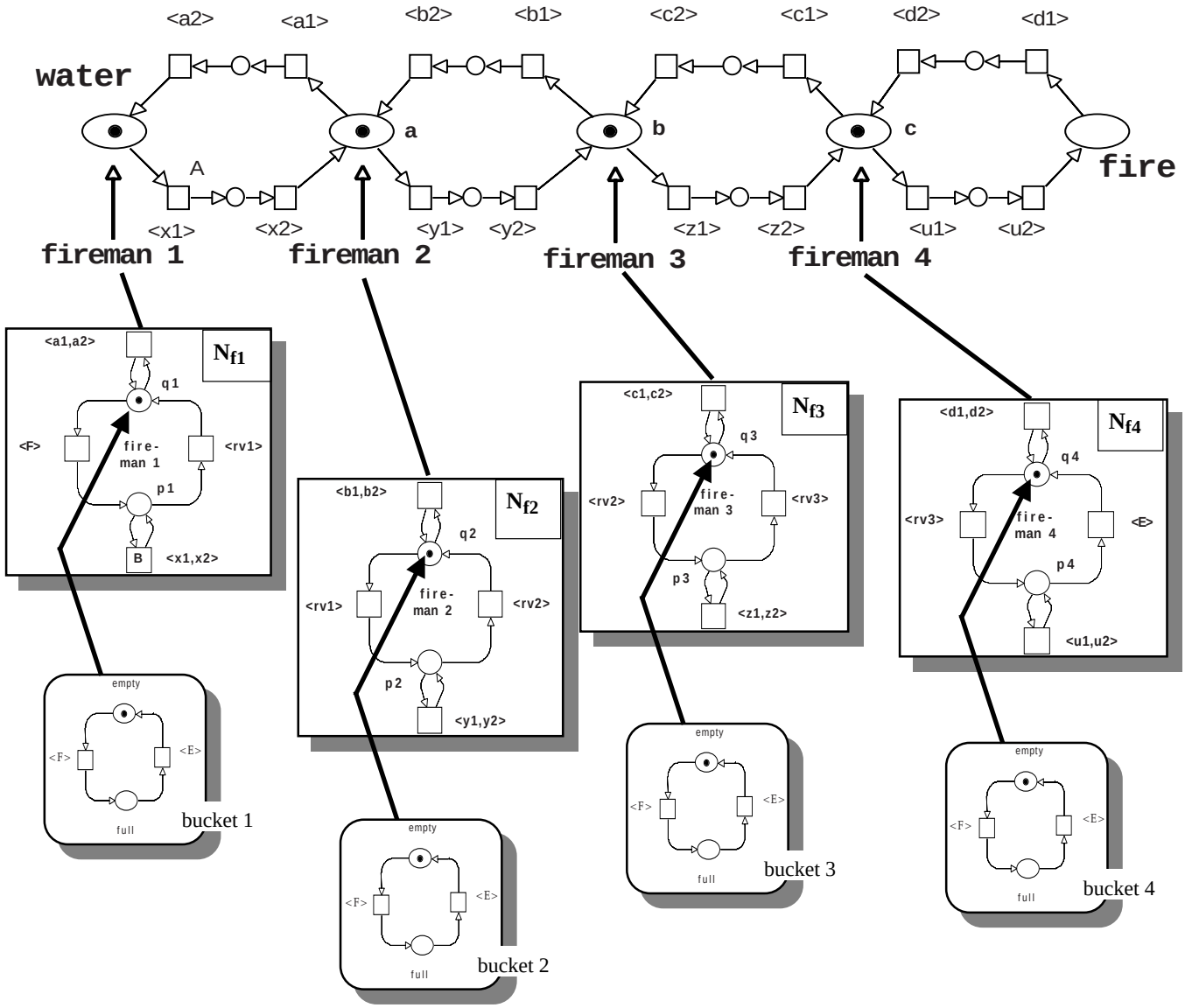


Fig. 2: Object system of Petri's fire extinction

men proceed on their routes interacting with the system net.

The interaction labelled <rv1> in the object nets N_{f1} and N_{f2}, however, describes that firemen 1 and 2 meet and perform the common action of "bucket exchange", a kind of rendez-vous. Such common action is possible if (and only if) they are in the same place, which is the rendez-vous-point "a" in our example. To keep the approach simple, a swapping of buckets between firemen is not formally represented in this paper. An extension in this direction is quite obvious, however.

To obtain a 3-level hierarchy, we add the object level of the buckets. The token in the fireman object net N_{fi} is to be understood as an object net "bucket i" at the bottom of fig. 2 . Buckets are in either the states „full“ or „empty“. The transitions marked <F> (fill) and <E> (empty) interact with corresponding transitions in firemen 1 and 4.

The behaviour of this object net is rather complex. We believe, however, that it is more natural than a "flat" modelling with only one object level, as shown in fig. 3. This net is constructed by merging interacting transitions and is called a " one-level equivalent". Here the object levels are hard to distinguish (though the third level of buckets is not shown). By this example it becomes apparent that object systems can be used as a kind of hierarchy modelling. New levels of abstractions (like the bucket level in the example) can be easily added without much affecting the other levels. The example of fire extinction is not formally treated in this paper. It serves for motivating the direction of research. The formal results of this paper refer to a simpler model that will be introduced in section 3.

How to define concurrency in object systems? Obviously different objects in separated places may act concurrently. However, there is a second type of concurrency. An object may split up in different copies to perform tasks in parallel and then rejoin to use the outcome of the concurrent steps. Examples are agents sending copies to different sites or tasks split up in subtasks to be performed in different funcional units.

Fig. 4 gives a simpler example. The occurrence of transition t_1 of the system net SN should coincide with e_1 in the object net ON by interaction i_1. Then we observe some concurrent behaviour ending with a kind of "join" operation by 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 .

From the presented examples the reader may guess that for object systems there are many open questions such as:

- How to formally define object systems?
- What is the exact setting of the occurrence rule?
- What is a natural notion of a process?
- What is the semantics of autonomous actions?

Some of these questions will be treated in the following sections. This section will end with a first definition of an object system which may be understood as a general frame

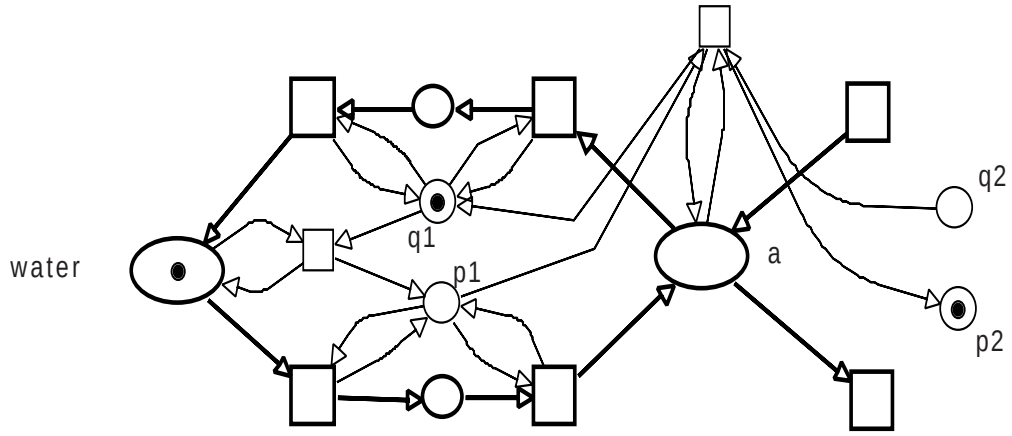
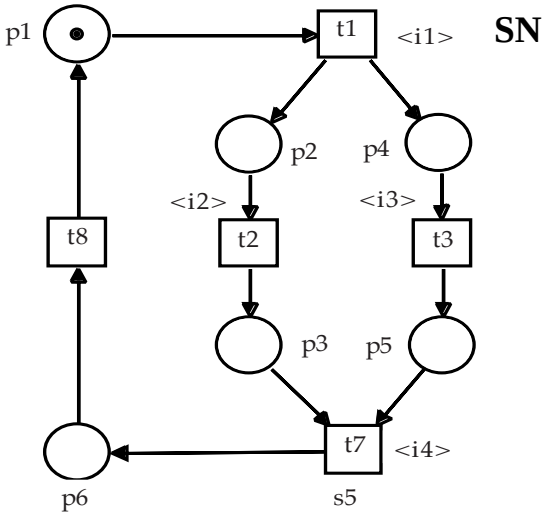


Fig. 3: One-level equivalent of fig. 2.2 (piece cut out)



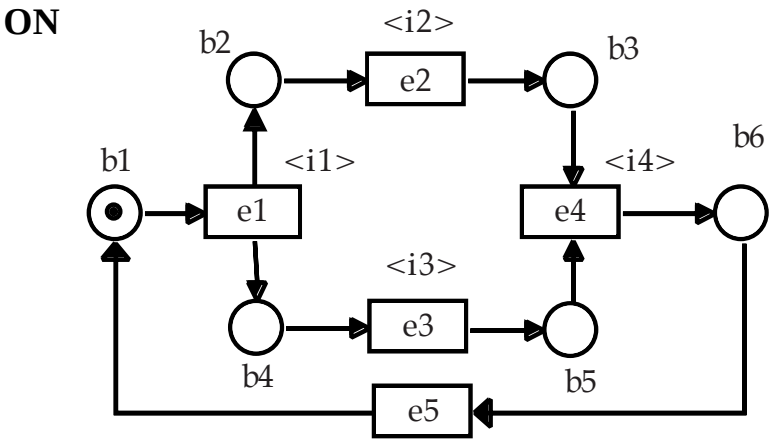


Fig. 4: Elementary object system "con-tasks"

for particular models.

Definition 2.1

An *object system* is a tuple OS = (SN,ON,,ρ) where

SN = (P,T,Col,Σ,I-,I+,M_o) is a Coloured Petri Net, called *system net*,

ON = {ON_i | 1≤i≤n}, is a finite set of (pairwise disjoint) Coloured Petri Nets, called the *object nets* and

$\rho = \rho_1 \cup \rho_2$ is the interaction relation where

$\rho_1 \subseteq T \times E$ is the system/object interaction relation and

$\rho_2 \subseteq E \times E$ is the object/object interaction relation .

E is the (disjoint) union of all transitions of the object nets.

The formal description of object systems is rather complex. As many important features can be studied using simpler subclasses, in this report we restrict object nets ON_i (1≤i≤n) to Elementary Net Systems. Such object systems will be called *object-elementary*. Moreover, Elementary Object Systems, as introduced in chapter 3, are restricted to only one object net (i.e. n=1) that may, however, split up in different copies operating concurrently.

Definition 2.2

An object system OS = (SN,ON,,ρ) with system net SN = (P,T,Col,Σ,I-,I+,Mo) is called *object-elementary* if

- the object nets ON_i (1≤i≤n) are Elementary Net Systems ON_i = (B_i,E_i,F_i,m_{oi}) (cf appendix),
- the set Σ of colours contains colours $\mathbf{col_i}$:= {(ON_i,proc) | proc∈PROC(ON_i)} (1≤i≤n), i.e. the object net ON_i in all its processes (cf appendix),
- all object nets in the initial marking (process) M_o of the system net SN are of the form (ON_i,m_{oi}), i.e. they are in the initial marking (or process) m_{oi} themselves.

For didactical reasons a full definition and in particular the semantics of object systems is not given here. In the next section, however, fundamental properties of such a semantics are studied using a more restricted (and simpler) model, where also the system net is an Elementary Net System. In particular, the choice of processes instead of markings in the preceding definition of $\mathbf{col_i}$ will be motivated.

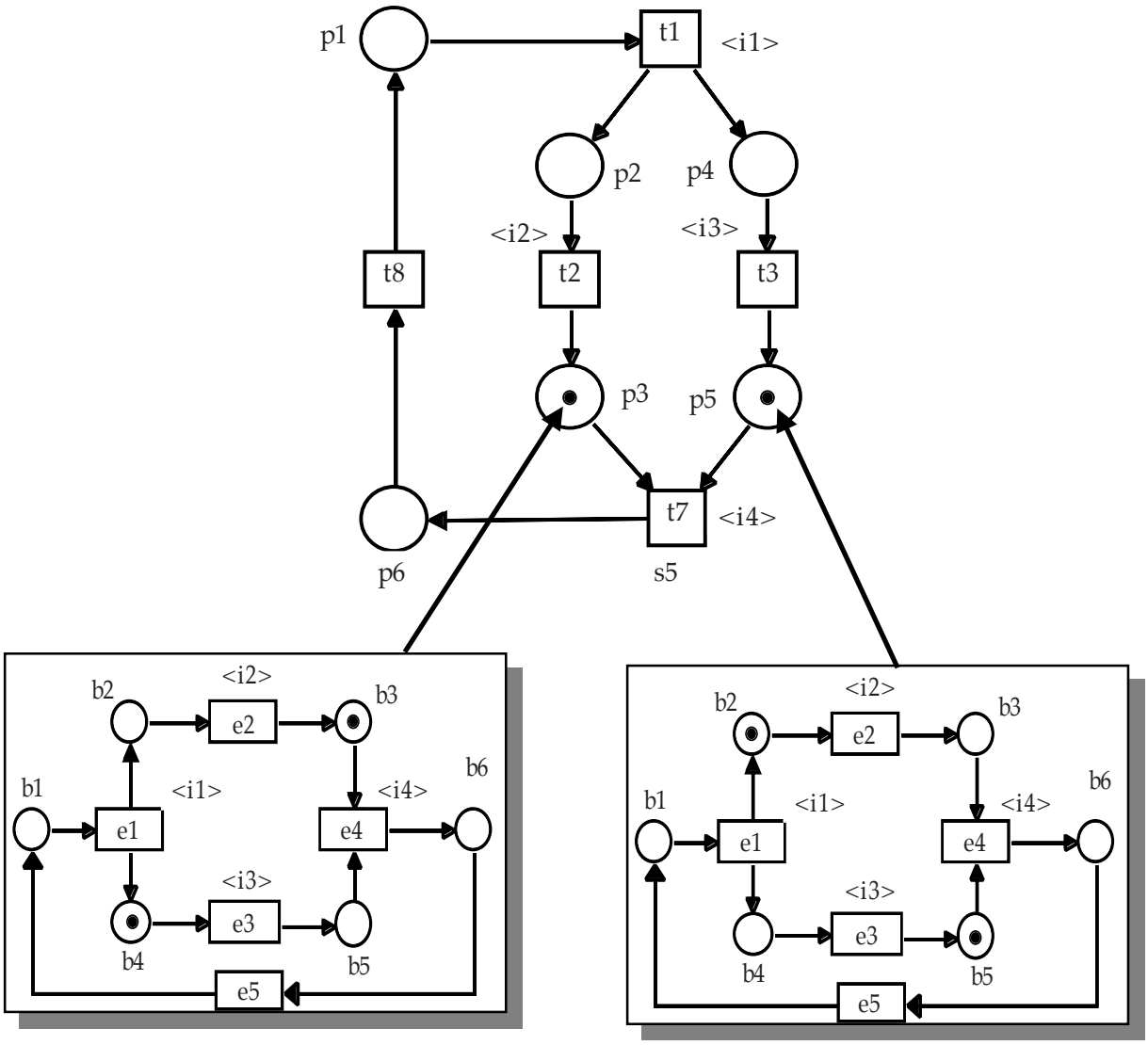


Fig. 5: Follower marking of fig. 4

3. Elementary Object Systems

The object system of fig. 4 has the particular property that both system net and object net are EN systems (Elementary Net Systems, see [Thiagarajan 87] and appendix A). This restriction allows clear definitions and the study of elementary properties.

Definition 3.1

An *elementary object system* is a tuple EOS = (SN,ON,ρ) where

SN= (P,T,W,M_o) is an EN system, 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 (cf appendix).

Fig. 4 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. 6 (i_1 is removed to illustrate autonomous transitions), but with a different system net. By this system net the "parallel" transitions e_1 and e_2 perform in a serial way. Since the system net is a state machine, the object system is simple.

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.

Definition 3.2

A *bi-marking* of an 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 $t\rho\rho = \emptyset$ 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 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

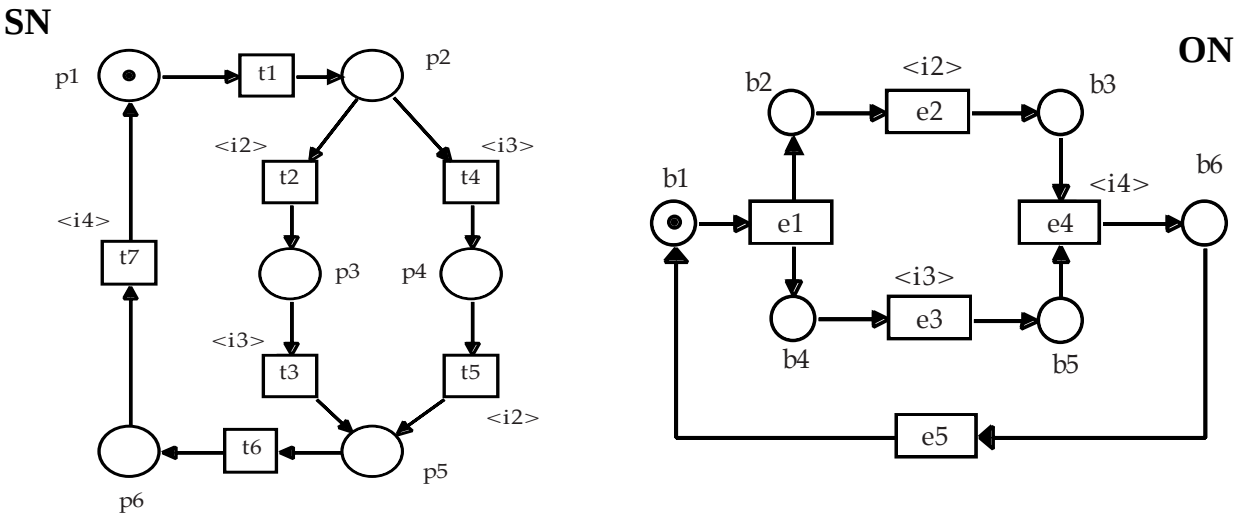


Fig. 6: Simple elementary object system "ser-task"

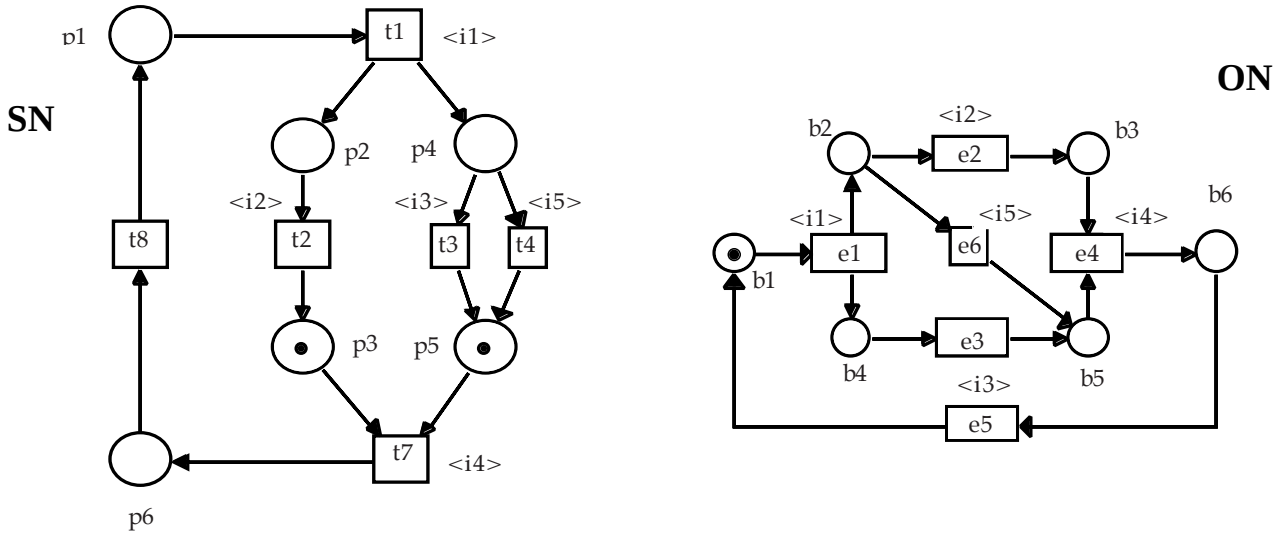


Fig. 7: Counterexample

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 6, for example, we obtain the following sequence: $[\lambda,e_1]$, $[t_1,\lambda]$, $[t_4,e_3]$, $[t_5,e_2]$, $[t_6,\lambda]$, $[t_7,e_4]$, $[e_5,\lambda]$. 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 sematics 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. 4 again in a bi-marking $(M,m) = (\{p_3,p_5\},\{b_3,b_5\})$ 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 <u>different</u> copies of the object net, as graphically visualized in fig. 5! 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_5 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 .

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. 5). As the modified example of fig. 7 shows, this is not adequate either. After an occurrence sequence like $[t_1,e_1]$, $[t_2,e_2]$, $[t_4,e_6]$, where t_4 occurs instead of t_3, a state as shown in fig. 8 is reached. From the view of the actual state, t_7 "sees" conditions b_3 and b_5 holding in its input places and therefore "concludes" that $[t_7,e_4]$ can occur. It should be clear however, from an intuitive way that such an occurrence is not the right way to model a "join-action" corresponding to the "split-action" $[t_1,e_1]$, since e_3 did not occur. The reason for this wrong modelling is that the copies of the object nets in fig. 8 are "history-less", i.e. they do not distinguish the evolution of the object net process.

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 fig. 9 and 10 two alternative such process-oriented markings are shown. While the merge of the first is consistently extendable the second is not. As a consequence, we will define the merging of distributed object net processes.

In a marking of an elementary object system, a place may be empty or contain the object

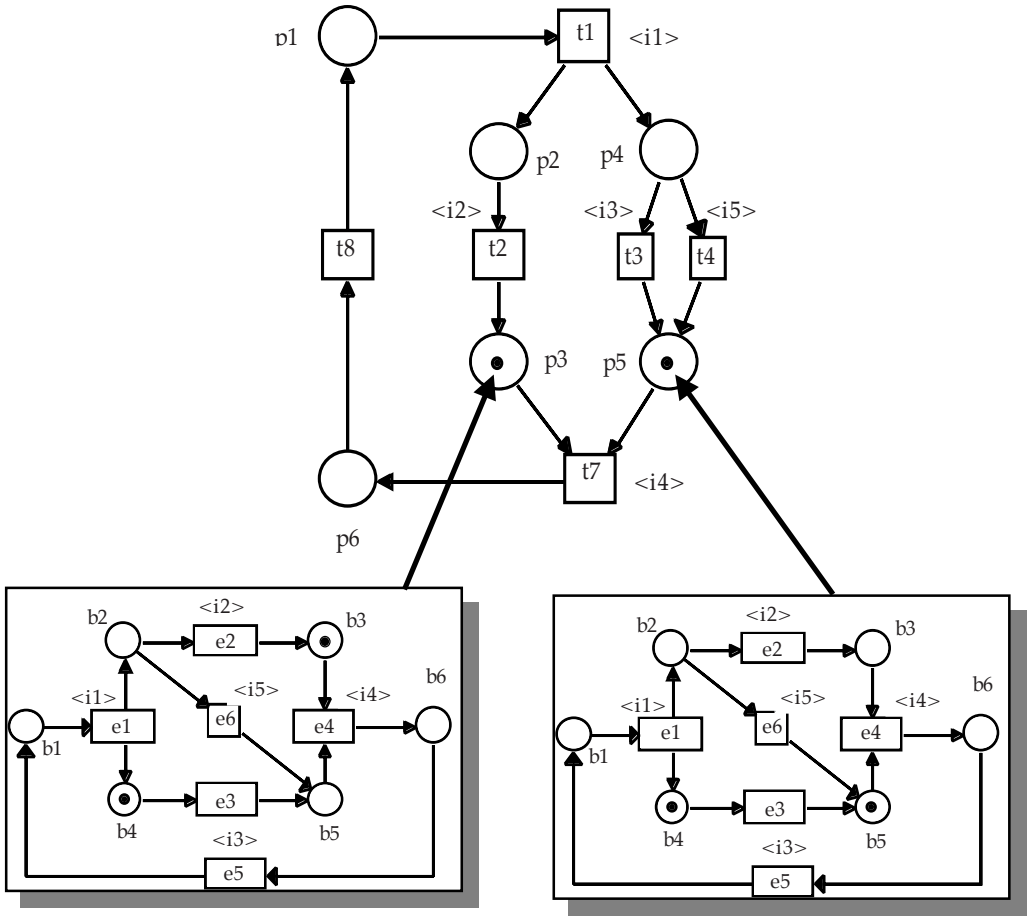


Fig. 8: Reachable state of the counterexample of fig. 7

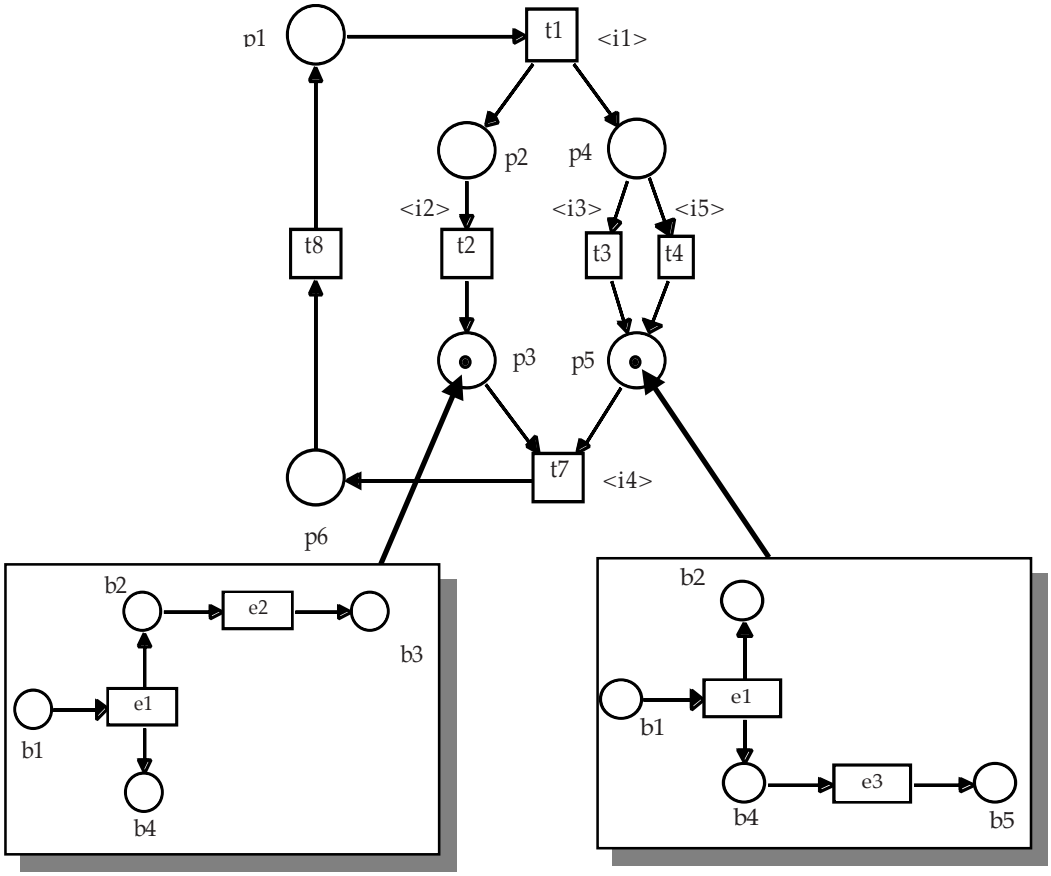


Fig.9: Two consistent ON-processes

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) (cf. appendix) 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 3.3

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.

For introducing the new occurrence rule, consider the EOS of fig. 4. 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. The preconditions of e4 are satisfied if all copies of the object net are taken into consideration. The joint information is obtained by the least upper bound lub of the processes in p_3 and p_5.

Definition 3.4

Let $\underline{M}$ be a p-marking of an 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 necessarry in any case that all input places $p \in {}^\bullet t$ are occupied, the process LUB := lub($\{\underline{M}(p) \mid p \in {}^\bullet t\}$) exists (cf. appendix A), and all output places $p \in t^\bullet$ are empty in $\underline{M}$.
In case
a) tρ=Ø we write $\underline{M} \rightarrow_{[t,\lambda]}$ and in case of
b) tρe the pair [t,e] is activated (denoted $\underline{M} \rightarrow_{[t,e]}$) if e is enabled for LUB
(cf. appendix A)
The follower p-marking $\underline{M}'$ is defined by

$\underline{M}'(p) = \emptyset$ if $p \in {}^\bullet t$,
$\underline{M}'(p)$ = LUB in case a) and $\underline{M}'(p)$ = LUB e in case b) if $p \in t^\bullet$, and
$\underline{M}'(p) = M(p)$ 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 $\underline{M} \rightarrow_{[t,e]} \underline{M}'$.

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 by

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

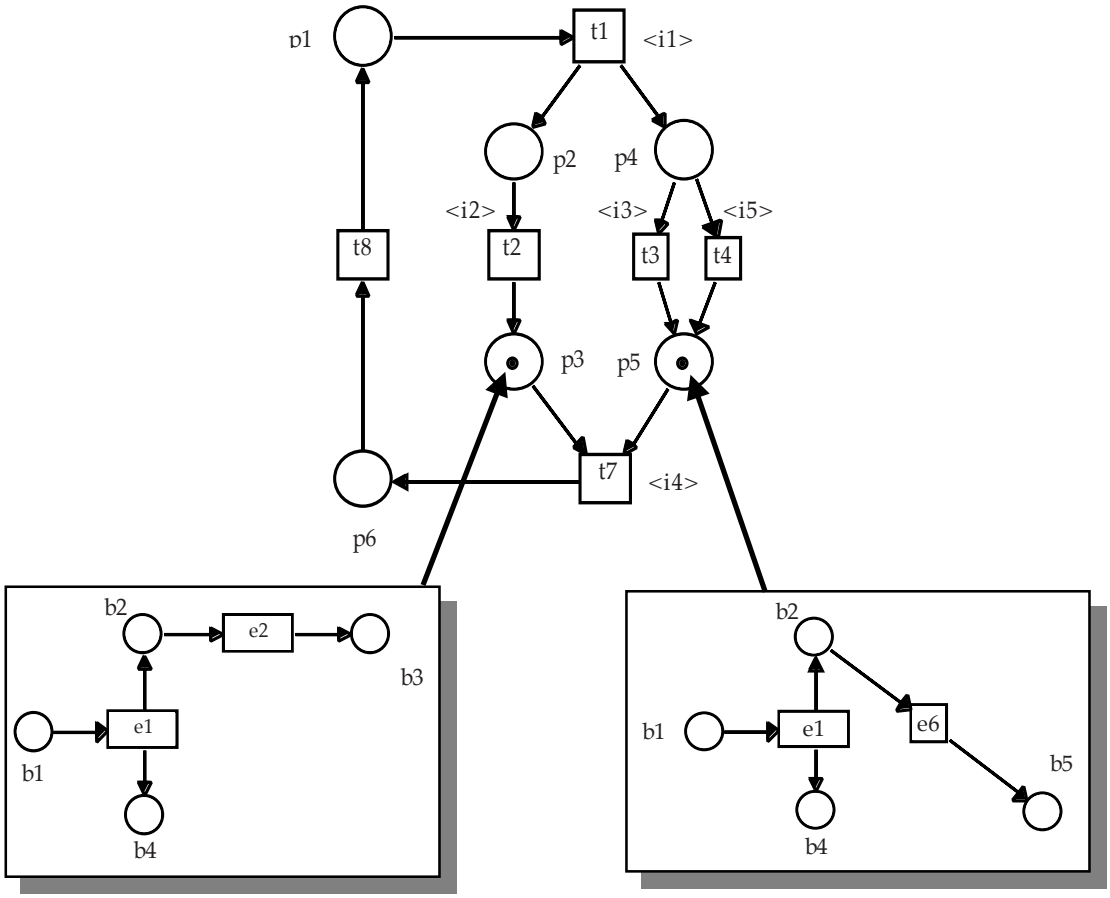


Fig. 10: Two inconsistent ON-processes

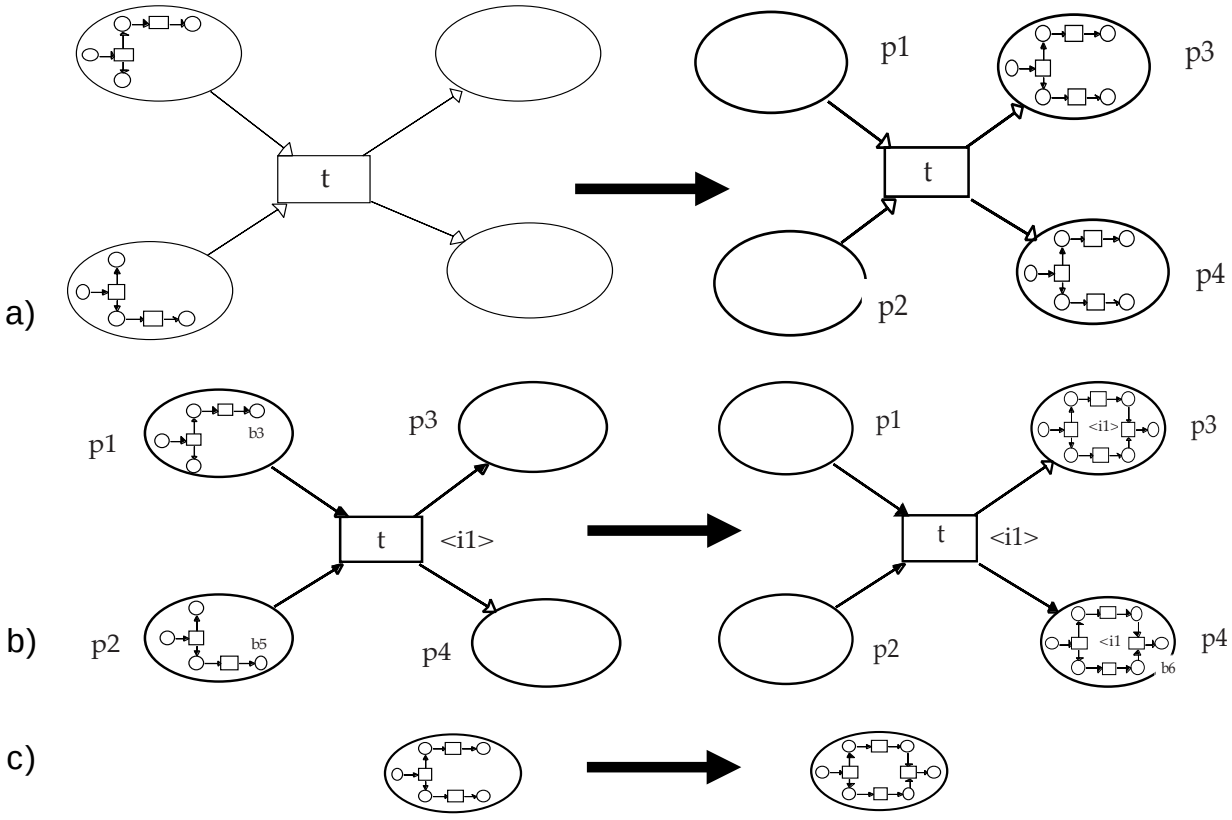


Fig.11: Object system occurrence rule : a) transport, b) interaction, and c) autonomous occurrence

Case c) is called an *object-autonomous* or an *autonomous* occurrence and is denoted by $\underline{M} \rightarrow_{[\lambda,e]} \underline{M}'$.
The initial marking is assumed to be $\underline{M}_o(p) = m_o$ iff $p \in M_o$.

In fig. 11 all cases a), b) and c) of the new 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 doesn't exist. Recall that for the p-marking of the object net from fig.9 case b) applies but not for the p-marking of fig.10.

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 3.5

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

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 $\varnothing$. (Note that in this context m_o means the initial process of EOS). $FS(EOS) := \{ w \in Q^* \mid \exists \underline{M}: \underline{M}_o \rightarrow_w \underline{M} \}$ is the *set of occurrence sequences* of EOS, and $R(EOS) := \{\underline{M} \mid \exists\ w : M_o \rightarrow_w \underline{M}\}$ is the set of *reachable p-markings*, also called the *reachability set* of N.

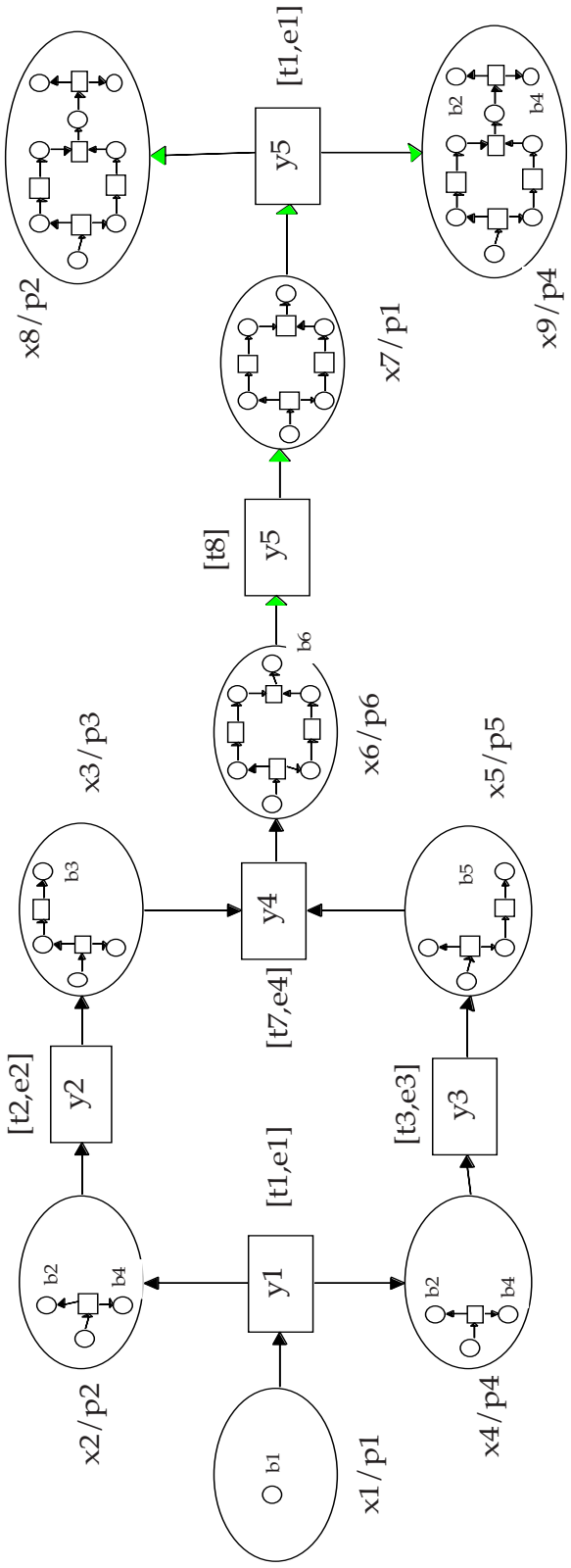


Fig. 12: A process of the object system "Con-Tasks" of fig. 4

4. Processes of Elementary Object Systems

The definition of processes of elementary object systems is quite obvious if autonomous occurrences are not considered. To give an example, in fig. 12 a process of the elementary object system "con-tasks" EOS = (SN,ON,ρ) from fig. 4 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.

Concurrency of transitions is a fundamental topic of Elementary Net Systems. Transitions may occur concurrently if their input and output places are disjoint. This generalizes to autonomous transitions of object systems and is formally stated in lemma 4.1.a). 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 4.1.b) and graphically represented in fig. 14 for the case n=1 and m=2.

Lemma 4.1

Let EOS = (SN,ON) be an elementary object system and t a system-autonomous transition of SN and e an object-autonomous transition of ON. Suppose $|{}^\bullet t| = m \geq 1$ and $|t^\bullet| = n \geq 1$. Let t be enabled in a p-marking $\underline{M}$.

a) If e is enabled in some proc = $\underline{M}(p)$ such that $p \notin {}^\bullet t$ then for all $\underline{M}'$ we have
$\underline{M} \rightarrow_v \underline{M}'$ where $v=[t,\lambda][\lambda,e]$ iff $\underline{M} \rightarrow_w \underline{M}'$ where $w=[\lambda,e][t,\lambda]$.

b) If e is enabled in proc = $\underline{M}(p)$ for all $p \in {}^\bullet t$, then for all $\underline{M}'$ we have
$\underline{M} \rightarrow_v \underline{M}'$ where $v=[t,\lambda][\lambda,e]^n$ iff $\underline{M} \rightarrow_w \underline{M}'$ where $w=[\lambda,e]^m[t,\lambda]$.

Proof:

a) Since t is enabled, all $p' \in t^\bullet$ are empty, hence $p \notin t^\bullet$. If $\underline{M} \rightarrow_{[t,\lambda]} \underline{M}''$, then $\underline{M}''(p)=\underline{M}(p)$ since $p \notin {}^\bullet t$ and $p \notin t^\bullet$.Therefore $\underline{M} \rightarrow_{[\lambda,e]} \underline{M}'''$ and $\underline{M}''' \rightarrow_{[t,\lambda]} \underline{M}'$ and $\underline{M} \rightarrow_w \underline{M}'$. The reverse direction is proved similarly.

b) If $\underline{M} \rightarrow_{[t,\lambda]} \underline{M}''$, then all input places of t contain the same process proc in $\underline{M}$. Since proc is not changed proc is contained in all output places of t in the follower marking $\underline{M}''$ where e is enabled by assumption. After the occurrence of $[\lambda,e]^n$ all output places of t contain proc e in $\underline{M}'$. (For the notation see appendix A.) Hence e is also enabled in all of $m = |{}^\bullet t|$ input places w.r.t. the marking $\underline{M}$. For $w' := [\lambda,e]^m$ we then have $\underline{M} \rightarrow_{w'} \underline{M}'''$, and since all input places contain proc e in $\underline{M}'''$ after the occurrence of $[t,\lambda]$, the process proc e is in all ontput places of t. Thus we have reached the same marking $\underline{M}'$.

Conversely, if $\underline{M} \rightarrow_w \underline{M}'$ where $w=[\lambda,e]^m[t,\lambda]$ then there is some intermediate marking $\underline{M}''$ such that for $w' = [\lambda,e]^m$ we have $\underline{M} \rightarrow_{w'} \underline{M}''$. Since $[t,\lambda]$ is enabled in $\underline{M}''$, all input places contain the same process proc e for some proc. Hence all input places of t contain proc in $\underline{M}$ and t is enabled in $\underline{M}$. After the occurrence of $[t,\lambda]$ all $n = |t^\bullet|$ output places of t contain proc where e is enabled. As a consequence $[\lambda,e]^n$ leads to the marking $\underline{M}'$. q

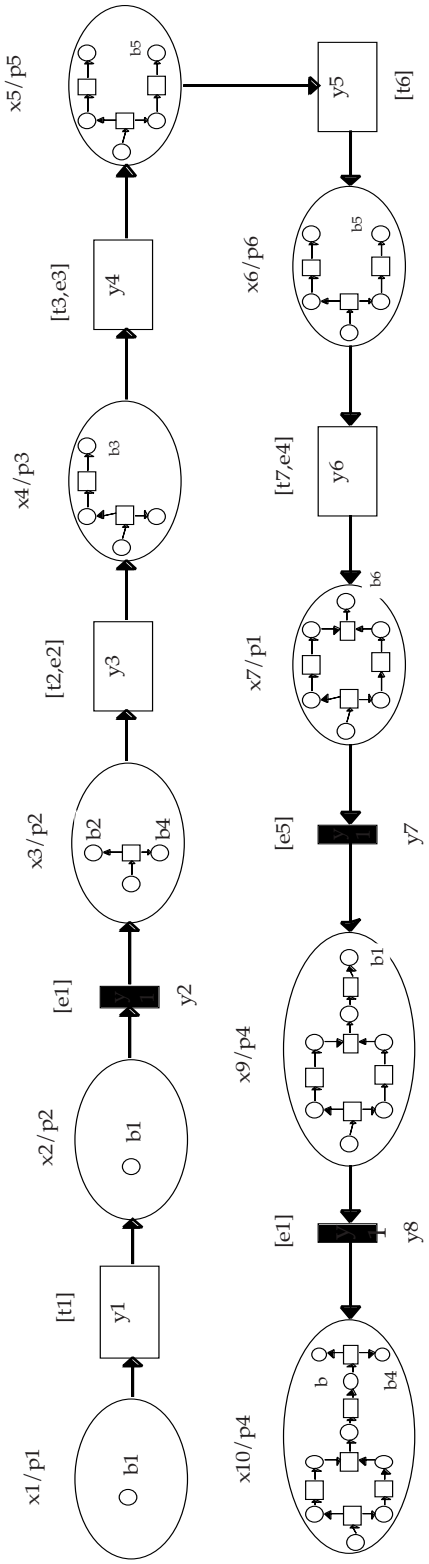


Fig. 13: A process of the object system "Ser-Task" of fig. 6

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. 14. Furthermore it is straight-forward to introduce an equivalent step containing both transitions as in fig. 14 c). We will call this a reduced process equivalent. Next we formally define a process of an elementary object system.

Definition 4.2

For a given firing sequence $w \in FS(OS)$ of an elementary object system $EOS = (SN,ON,_\rho)$, 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 $\phi : \underline{P}_\pi \cup \underline{A}_\pi \rightarrow P \cup T$ and $\mu : \underline{P}_\pi \rightarrow PROC(ON)$. $\underline{A}_\pi \subseteqq \underline{T}_\pi$ is the subset of non-autonomous transitions. proc(w) is defined by induction over Q* (as well as $\underline{A}_\pi$). Furthermore for each object net process $proc_2= (B_\pi,E_\pi,F_{2\pi},\phi_2) = \mu(p)$, $p \in \underline{P}_\pi$ an *extended interaction relation* $\rho_\pi \subseteq \underline{T}_\pi \times E_\pi$ is defined.

I. If $w = \lambda$, then $\underline{P}_\pi = \{p_\pi \mid p \in M_o\}$ with $\phi(p_\pi) = p$ and $\mu(p_\pi) = mo$ for all $p_\pi \in P$. $\underline{A}_\pi$ and ρ_π are empty.
(Note: markings are interpreted here as processes in the form of an initial process (see appendix).

II. Let be $\underline{M}_o \rightarrow_w \underline{M} \rightarrow_{[x,y]} \underline{M'}$ and $proc(w) = (\underline{P}_\pi,\underline{T}_\pi,\underline{F}_\pi,\phi,\mu)$ be the process of w . Then for $[x,y] \in Q$ we define $proc(w[x,y]) = (\underline{P'}_\pi,\underline{T'}_\pi,\underline{F'}_\pi,\phi',\mu')$ for each of the cases a), b) and c) of definition 3.4.:

a) If $[x,y] = [t,\lambda]$ and $t\rho=\varnothing$, then there is a subset $P_1 \subseteq \underline{P}_\pi$ having no output transitions (i.e. $\underline{P}_1^\bullet = \varnothing$) such that $\phi(\underline{P}_1) = {}^\bullet t$. By the enabeling rule all places p in $\underline{P}_1$ contain a process $\mu(p)=proc_1$ such that their least upper bound LUB:=lub$\{\mu(p) \mid p \in \underline{P}_1\}$ exists. To obtain $(P_\pi',T_\pi',T_\pi',\phi', \mu')$ 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 t' with $\phi'(t') = t$ to $\underline{T}_\pi$ (i.e. $\underline{T}_\pi' := \underline{T}_\pi \cup \{t'\}$).

a_3) Add arcs from P_1 to t' and from t' to P_2 (i.e.: $\underline{F}_\pi' := \underline{F}_\pi \cup \{ (p,t') \mid p \in P_1 \} \cup \{(t',p) \mid p \in P_2\}$).

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 $\mu'(p) = \mu(p)$ for the old places $p \in \underline{P}_\pi$ and for the new places $p_2 \in P_2$ with $\phi(p_2) \in t^\bullet$ we define $\mu'(p_2) :=$ LUB (i.e. the output places of t contain the same process LUB. $\underline{A}_\pi$ and ρ_π are extended to LUB.)

b) If $[x,y] = [t,e]$ 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 $\mu'(p) = \mu(p)$ for the old places $p \in P_\pi$ and for the new places $p_2 \in P_2$ with

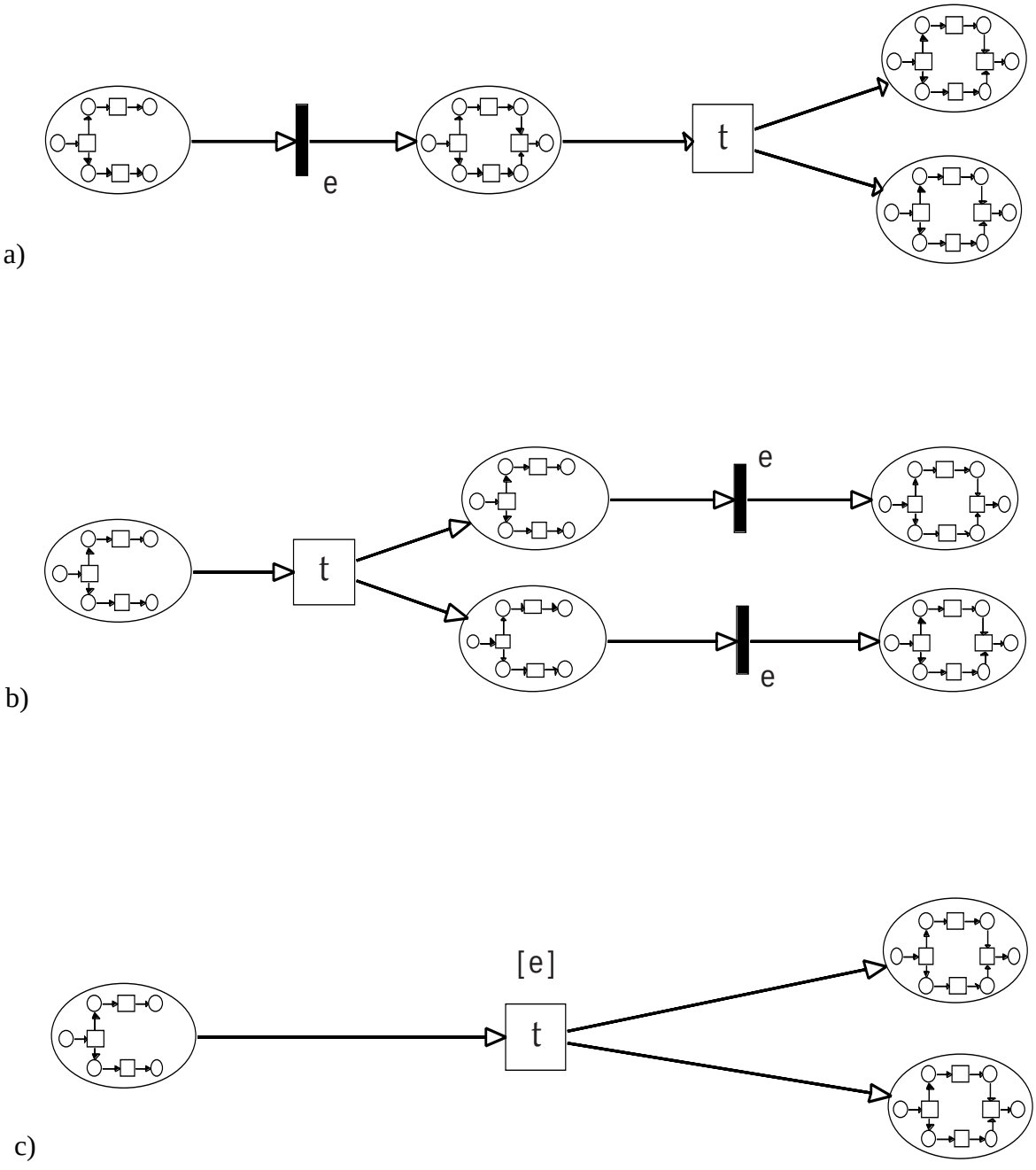


Fig. 14: Concurrent autonomous transitions

$\phi(p_2) \in t^\bullet$ we define $\mu'(p_2)$:=LUB e (i.e. the output places of t contain the process LUB extended by "e".).

$\underline{A}_\pi$ and ρ_π are extended to LUB and t' and (t,e) are added to $\underline{A}_\pi$ and ρ_π, respectively.

c) If [x,y] = [λ,e] and $\rho e = \emptyset$, then there is a place $p_1 \in P_\pi$ with $p1^\bullet = \emptyset$ such that "e" is enabled in the process $proc_1 = \mu(p_1)$.

c_1) Add a new place p_2 to $\underline{P}_\pi$ (i.e. $\underline{P}_\pi' = \underline{P}_\pi \cup \{p_2\}$).
c_2) Add a new transition t' with $t' \notin \underline{A}_\pi$ to $\underline{T}_\pi$ (i.e. $\underline{T}_\pi' := \underline{T}_\pi \cup \{t'\}$).
c_3) Add arcs from p_1 to t' and from t' to p_2 (i.e.: $\underline{F}_\pi' := \underline{F}_\pi \cup \{ (p_1,t'), (t',p_2)\}$).
c_4) Define $\phi' = \phi$ for all old places and $\phi'(p_2)=\phi(p_1)$.
c_5) Define $\mu'(p) = \mu(p)$ for the old places $p \in \underline{P}_\pi$ and for the new place $p_2 \in P_2$ we define $\mu'(p_2) := proc_1$ e (i.e. the output places of t' contains the same processes as the input places but extended by e). $\underline{A}_\pi$ and ρ_π are not modified.

Examples of object system processes are shown in figures 12 and 13. 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)). y_1,y_2, ... denote transitions.They have additional inscriptions of the form

a) $[t_i]$ if ti is a transport, i.e. $t_i\rho=\emptyset$,
b) $[t_i,e_j] \in \underline{T}_\pi \times E$ if $\phi(t_i)$ interacts with e_j, i.e. $\phi(t_i)\rho e_j$ (or $t_i \rho_\pi e_j$) and
c) $[e_j]$, $e_j \in E$ if e_j is autonomous i.e. $e_j\rho_\pi=\emptyset$.

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

Lemma 4.3

With the notation of def. 4.2 the following holds: for each place $p \in \underline{P}_\pi$, $proc_2 = (B_\pi,E_\pi,F_{2\pi},\phi_2) \in \mu(p)$ and $e \in E_\pi$ with $\rho_\pi e \neq \emptyset$ there is some transition $t <_{proc(w)} p$ (i.e. "before" p) such that $t\rho_\pi e$.

Proof: e is either introduced to $proc_2$ in μ(p) in step II b) of definition 4.2 (then $t\rho_\pi e$ for some $t \in {}^\bullet p$) or e is created with a copy of $proc_2$ from some $p_1 \in {}^\bullet t$ in one of the other steps (in that case the statement holds by induction). q

Definition 4.4

An autonomous transition t of an object system process as introduced in definition 4.2 c) has a unique input place p_1 and a unique output place p_2 with $\phi(p_1) = \phi(p_2)$. Therefore identifying p_1 and p_2 and eliminating t gives a consistent notion of a process. For the merged place p the contained process μ(p) is defined either by $\phi(p_1)$ or $\phi(p_2)$. 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. They appear as to be coincident with interactive transition occur-

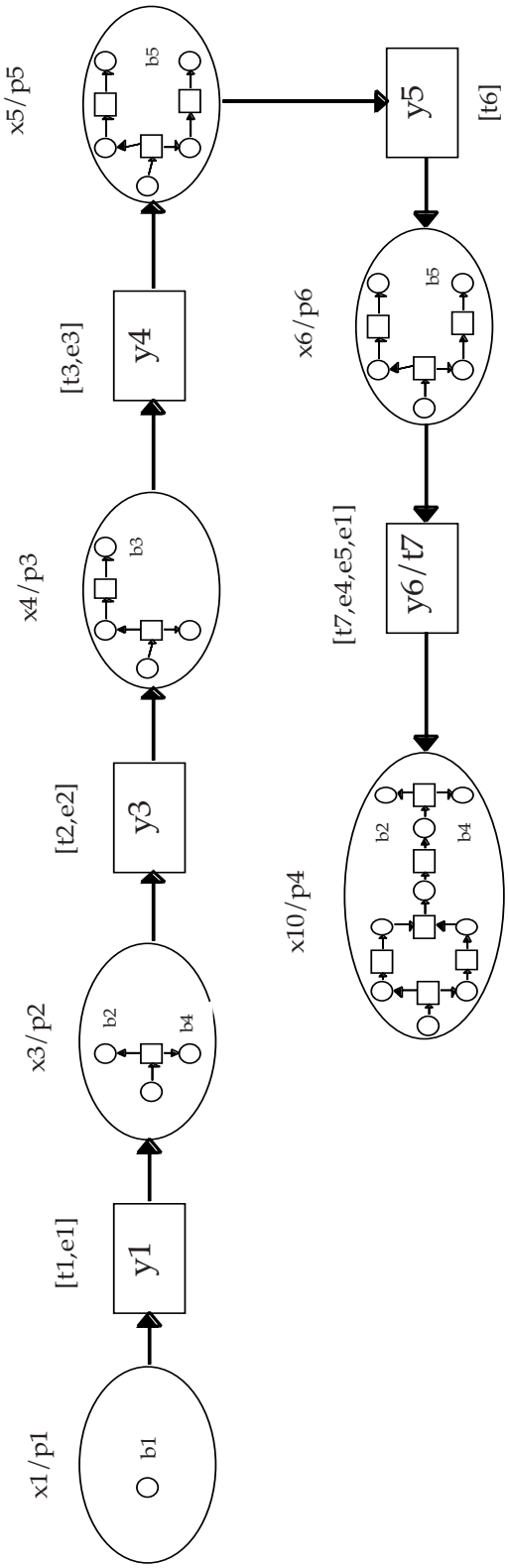


Fig. 15: A reduced form of the process of fig. 13

rences that occur just before or after. Fig. 15 shows a reduced form of the process from fig. 13. The first occurrence of e_1 appears to be coincident with t_1, whereas the terminal occurrences of e_5 and e_1 are merged with t_7. Since we have not formalized this part of the process representation, we choose to denote these steps by $[t_1/e_1]$ and $[t_7,e_4/e_5,e_1]$, respectively.

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 following theorem 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 cooperating processes. This is done in fig. 16. for the EOS "con-tasks" (fig. 4) and its process (fig.12). This figure contains on the top half 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 4.5

Let EOS = (SN,ON,ρ) be an elementary object system with SN= (P,T,W,M_o) and ON = (B,E,F,m_o). A *cop-process* (process in cooperating process form) of EOS is defined as a tripel

$\Theta = (proc_1,proc_2,\rho_\pi)$ where

proc1= $(P_\pi,T_\pi,F_{1\pi},\phi_1) \in PROC(SN)$

proc2= $(B_\pi,E_\pi,F_{2\pi},\phi_2) \in PROC(ON)$

To simplify the definition, we start from an EOS-process proc(w)= $(\underline{P}_\pi,\underline{T}_\pi,\underline{F}_\pi,\phi,\mu)$ in reduced form, as introduced in def. 4.2 and 4.4, having a latest place p_ω (cf. appendix).
Then we define

$proc_1 := (\underline{P}_\pi,\underline{T}_\pi,\underline{F}_\pi,\phi)$ (i.e. proc(w) without the inscriptions given by μ) and

$proc_2 := \mu(p_\omega)$ (i.e. the object net process in the latest place p_ω)

$\rho_\pi \subseteq T_\pi \times E_\pi$ is the relation from definition 4.2.

In fig. 16 the result of this construction is given when applied to the process of fig. 12 if y_5 ,x_8 and x_9 are deleted. Then x_7 is the requied latest place. The definition can be extended to the process of fig. 12 if the object net processes in the terminal cut (cf. appendix) of proc(w) the least upper bound LUB exists.

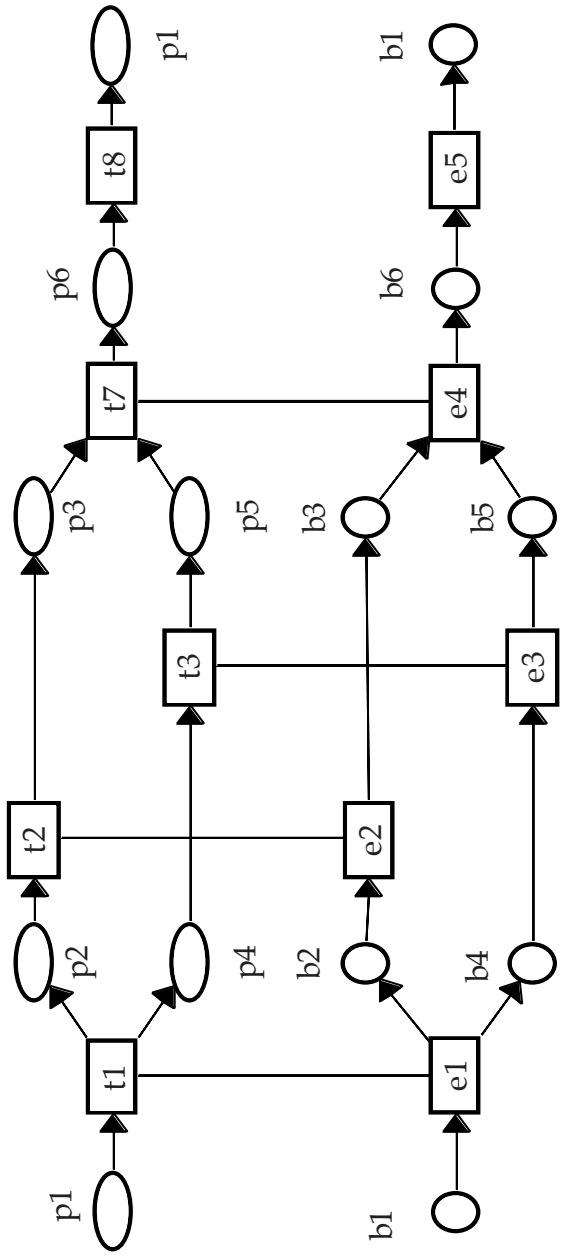


Fig. 16: Cooperating process representation of the process of fig.12

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 subnet of $proc_2$ "generated" by T_1.

Lemma 4.6

a) Given a cop-process $\Theta = (proc_1, proc_2, \rho_\pi)$ of an EOS = (SN,ON,ρ) then
$\forall\, t_1 \in T_\pi \;\; \forall\, e_1, e_2 \in E_\pi : t_1 \rho_\pi e_1 \wedge t_1 \rho_\pi e_2 \;\Rightarrow\; e_1 = e_2$ holds.

b) There is a cop-process $\Theta = (proc_1, proc_2, \rho_\pi)$ of an EOS = (SN,ON,ρ) such that
$\forall\, t_1, t_2 \in T_\pi \;\; \forall\, e_1 \in E_\pi : t_1 \rho_\pi e_1 \wedge t_2 \rho_\pi e_1 \;\Rightarrow\; t_1 = t_2$ is <u>not</u> true in general.

Proof: In the construction of ρ_π each transition $t \in T_\pi$ appears only once, whereas $e \in E_\pi$ may appear in different copies. In fig.18 a cop-process of the EOS from fig.17 is shown together with the relation ρ_π. The cop-process fails to have property b). q

Definition 4.7

Let be $T_{int} := \{t \in T_\pi \mid t \rho_\pi \neq \emptyset\}$ 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. Then (by lemma 4.6 a)) $\varphi : T_{int} \rightarrow E_{int}$ with $\varphi(t) = e$ iff $t \rho_\pi e$ is a mapping. (By lemma 4.6b) φ may be non-injective.
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 4.3 can be rewritten as follows.

Lemma 4.8

Given a *cop-process* $\Theta = (proc_1, proc_2, \varphi)$, then $e_1 <_{proc_2} \varphi(t)$ implies $\exists t_1 : t_1 <_{proc_1} t \wedge \varphi(t_1) = e_1$.

Proof: By definition $e = \varphi(t)$ iff $t \rho_\pi e$ in the corresponding EOS-process "proc". By induction on the construction of proc transitions e_1 and e are in $\mu(p)$ for any $p \in t^\bullet$. By lemma 4.3 there is a transition $t_1 <_{proc} p$ with $t_1 \rho_\pi e$. By lemma 4.6 (since $t \rho_\pi e$) $t_1 \neq t$, hence $t_1 <_{proc} t <_{proc} p$, and also $t_1 <_{proc_1} t <_{proc_1} p$. q

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

Definition 4.9

Given an elementary object system EOS = (SN,ON,ρ) and processes
$proc_1 = (P_\pi, T_\pi, F_{1\pi}, \phi_1) \in PROC(SN)$,

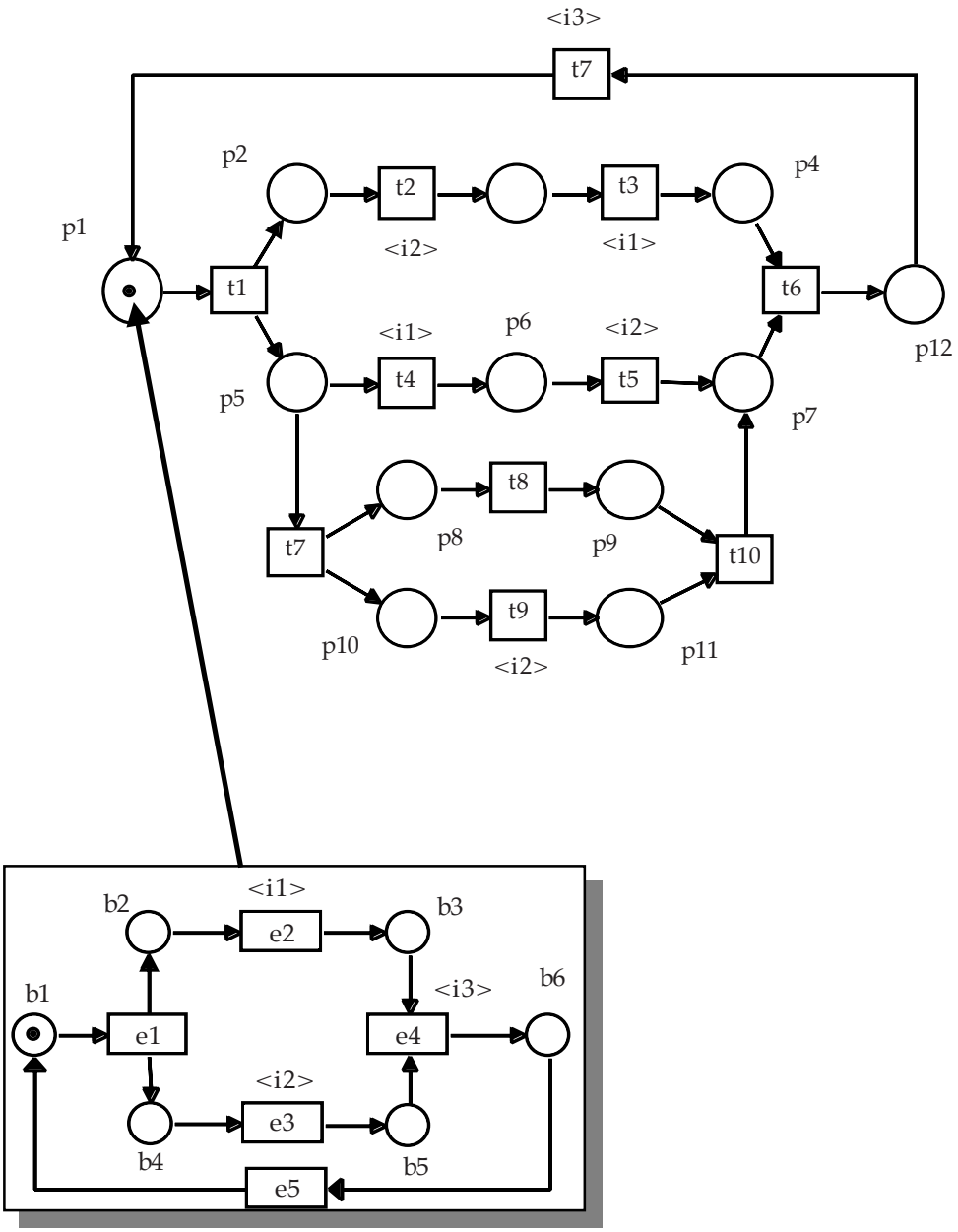


Fig. 17: EOS illustrating the theorem.

$proc_2 = (B_\pi, E_\pi, F_{2\pi}, \phi_2) \in$ PROC(ON) and a mapping

$\varphi: A \to E_\pi$ with $A \subseteq T_\pi$

φ is called *interaction true* or *true* if

$\phi_1(t)\ \rho\ \phi_2(\varphi(t)) \Leftrightarrow t \in A$,

$\phi_1(t)\ \rho = \emptyset \Leftrightarrow t \notin A$ and

$e\rho = \emptyset \Leftrightarrow e \notin \varphi(A)$

The triple $\Theta = (proc_1, proc_2, \varphi)$ has the *extended process morphism property* (EMP) iff: **EMP**.: $e_1 <\!\!\cdot_{proc_2} e_2 \wedge t_2 \in \varphi^{-1}(e_2) \Rightarrow \exists\, t_1 <_{proc_1} t_2 : \varphi(t_1) = e_1$.

($<\!\!\cdot_{proc_2} \subseteq <_{proc_2}$ denotes the immediate successor relation of $<_{proc_2}$)

If φ is an injection, then $\varphi^{-1} : E_{int} \to T_{int}$ is a T-morphism (cf. appendix). There is a convincing interpretation such that object net transitions are considered 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 (i.e. $\varphi(t_1)=e_1$, $\varphi(t_2)=e_2$ and $\neg(t_1 < t_2)$) since 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 sequentially executed but not vice versa. The violation of injectivity of the mapping φ in figure 18 is expressing a redundant execution: tasks e_2 and e_3 are both executed in concurrent "branches" of the system net process. At transition t_6, by joining together these branches, identical object net processes appear in the input places of t_6. This can be used for the design of fault tolerant systems.

Theorem 4.10

Let be given an elementary object system EOS = (SN,ON,ρ) and a tripel $\Theta = (proc_1, proc_2, \varphi)$, where $proc_1 \in$ PROC(SN), $proc_2 \in$ PROC(ON) are processes and $\varphi: A \to E_\pi$, $A \subseteq T_\pi$ 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 4.8.

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 $\varphi: T_\pi \to E_\pi$ is a true mapping satisfying the EMP.

First we have to find a mapping $\mu: P_\pi \to$ PROC(ON) such that $proc = (P_\pi, T_\pi, F_\pi, \phi_1, \mu)$ is an EOS-process. This is done by defining

$$\mu(p) := past_{proc_2}(\{\varphi(t) \mid t <_{proc_1} p\}), \quad (*)$$

where $past_{proc_2}(A)$ is the subprocess generated by the set A (see appendix).

Next it must be shown that proc is an EOS-process in reduced form i.e. satisfying the occurrence rules of definitions 4.2 and 4.4. This is done by induction on the partial order $<_{proc}$ of proc.

I. If $^\bullet p = \emptyset$ then by the definition of "past" we have $\mu(p) = init(proc_2) = m_o$.

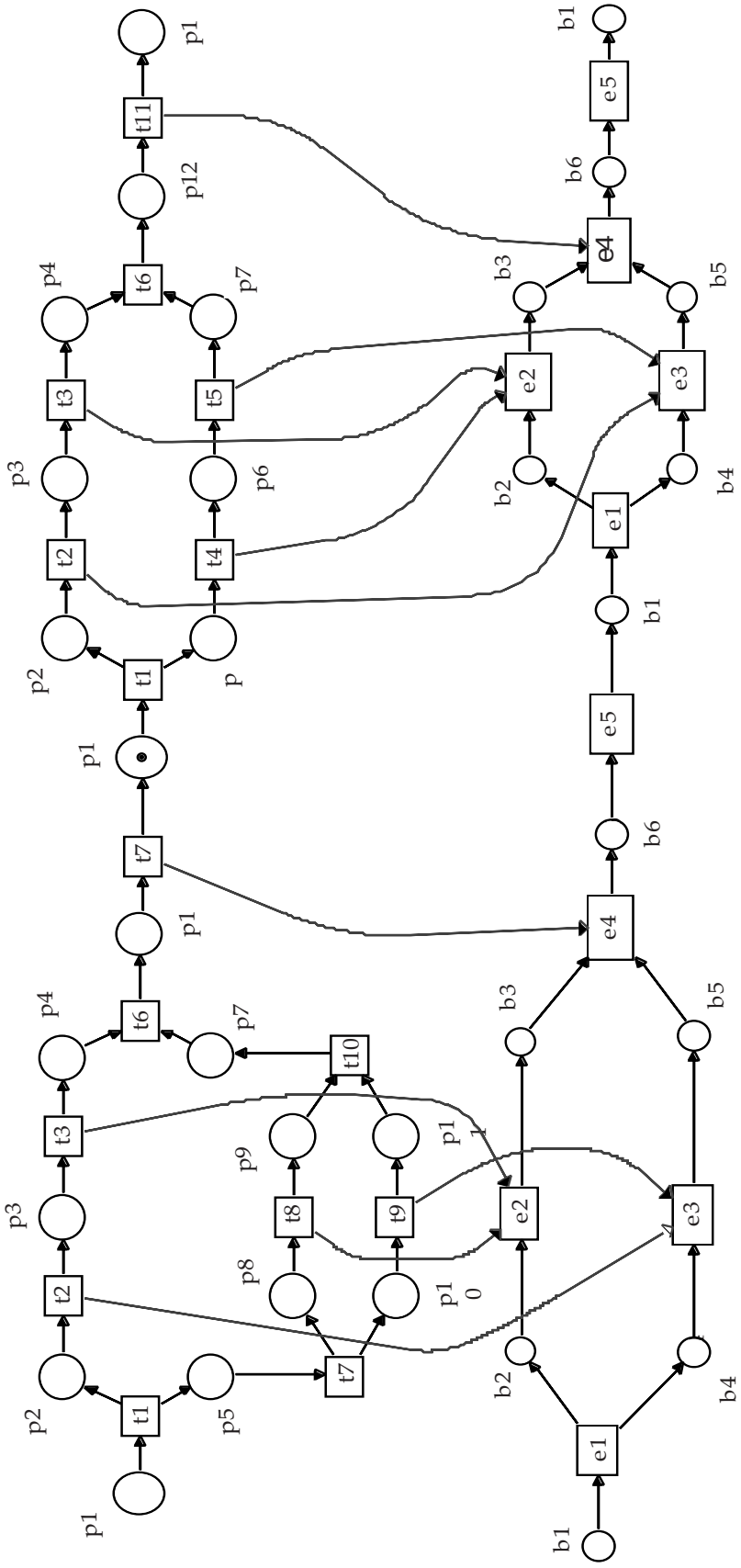


Fig. 18: Cop-process of the EOS of fig. 17 with mapping φ.

II. Let be $^\bullet p \neq \emptyset$ and $t \in {}^\bullet p$.

a) If $t \notin A$ then $\phi_1(t)\rho = \emptyset$, hence $\phi_1(t)$ is a transport. By (*) $\mu(p)$ is the least upper bound of all $\mu(p_1)$ from input places $p_1 \in {}^\bullet t$ that are well-defined by induction. Hence $\mu(p)$ is in accordance with definition 4.2 IIa.

b) If $t \in A$, then $\phi_1(t)\ \rho\ \phi_2(\varphi(t))$. Then for $e = \varphi(t)$ we have two subcases b_1 and b_2:
b_1) If $\exists\ e_1: e_1 <^\bullet_{proc_2} e$ does <u>not</u> hold, then by definition $\mu(p_1) = init(proc_2)$ for all $p_1 \in {}^\bullet t$ and t is enabled in EOS and $\mu(p) := past_{proc_2}(\{\varphi(t) |\ t <_{proc_1} p\}) = init(proc_2)$. Again, $\mu(p)$ is in accordance with definition 3.2 IIb).
b_2) If $\exists\ e_1: e_1 <^\bullet_{proc_2} e$ is true, then by the assumption EMP for each such e_1 there is some t_1 satisfying $t_1 <_{proc_1} t$ and $\varphi(t_1) = e_1$. Hence each predecessor transition e_1 of e is contained in a process of some input place of t. All these processes are well-defined by induction assumption and contained in their least upper bound. Hence, t is enabled and $\mu(p)$ is in accordance with def. 4.2b.

c) If $e \notin \varphi(A)$, then e is autonomous (i.e. $e\rho = \emptyset$) and either c_1 or c_2 holds as follows.

c_1) If $\exists e_1: e_1 \in \varphi(A) \wedge e <_{proc_2} e_1$ take the smallest such e_1. Then by definition of μ transition e is contained in $\mu(p)$ for any output place of $\varphi^{-1}(e_1)$, in accordance with the reduced form EOS-process definition.

c_2) If there is no $e_1 \in \varphi(A) \wedge e <_{proc_2} e_1$, by def. 4.4 transition e may not be contained in any process $\mu(p)$ in proc. q

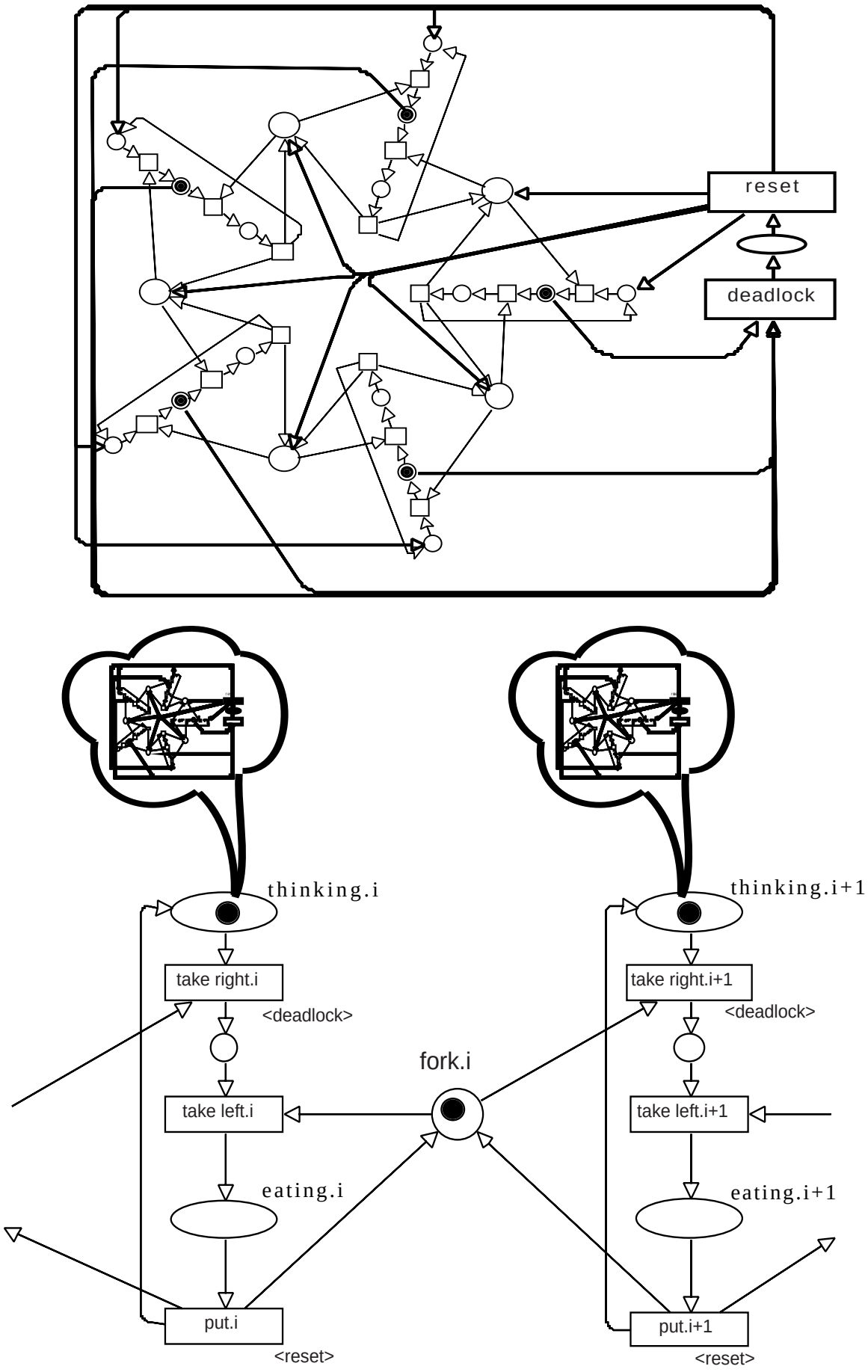


Fig. 19: Russian Dining Philosophers

5. Processes of Object Systems

Elementary object systems are restricted to contain a single object net (that may exist in several copies, however). In [Valk 95] a simplified version of the "Russian philosophers problem" due to Lakos [Lakos 94] was presented to give a simple example of an object system having several different object nets. The system net of this example is an ordinary five philosophers net containing transitions "take right.i", "take left.i" and "put.i" ($i \in \{1,..,5\}$) for managing the forks (see fig. 19 bottom).

Each of the five philosophers has a distinct instance ON_i ($1 \le i \le 5$) of the five philosophers net in his mind (fig. 19 top) and makes some mental simulation. Hence, in the beginning there are five object nets ON_i, each in the place "thinking.i" of the system net. Each of these ON_i is performing autonomous transition occurrences until it deadlocks. Then philosopher i stops thinking by the occurrence of a transition "take right.i". After eating and by the occurrence of a transition "put.i" he enters his thinking state again.To be able to think on his private philosophers problem again, the deadlock in his object net ON_i must be removed. To this end ON_i contains a transition "deadlock" that removes the tokens of the deadlock. Afterwards a transition "reset" is activated that restores the initial marking of ON_i. The interaction between the system net and the object net is defined in such a way that "take right.i" and "put.i" interact with "deadlock" and "reset", respectively. This object systems contains five different object nets. However they do not interact directly, but via sharing common resources, namely the forks.

To present an example with objects that interact directly the example of fire extinction was presented in section 2. Disregarding the bucket objects in figure 2, we obtain an object system containing four object nets. The interaction relation $\rho = \rho_1 \cup \rho_2$ (cf. def. 2.1) contains object/object pairs, as the transitions in N_{f1} and N_{f2} labelled <rv1>.

The occurrence rule of general object systems is not formally defined here, but should be thought of to be designed in the spirit of semantics of elementary object systems. In the following, for the case of object-elementary object systems (def. 2.2) the additional requirements for a formal semantics are discussed.

Recall definition 2.2 where $OS = (SN, ON, \rho)$ is an object-elementary object system, i.e.

- $SN = (P,T,Col,\Sigma,I-,I+,M_o)$ is a Coloured Petri Net, the system net,
- $ON = \{ON_i \mid 1 \le i \le n\}$, is a finite set of (pairwise disjoint) Elementary Net Systems
- $ON_i = (B_i,E_i,F_i,m_{oi})$ are the object nets and
- $\rho = \rho_1 \cup \rho_2$ is the interaction relation where
  - $\rho_1 \subseteq T \times E$ is the system/object interaction relation and
  - $\rho_2 \subseteq E \times E$ is the object/object interaction relation .
  - $E = \cup_{1 \le i \le n} E_i$ is the (disjoint) union of all transitions of the object nets.
- The set Σ of colours contains colours $col_i := \{(ON_i,proc)\ proc \in PROC(ON_i)\}$ $(1 \le i \le n)$, i.e. the object net ON_i in all its processes.

- All object nets in the initial marking (process) M_o of the system net SN are of the form (ON_i,m_{oi}), i.e. they are in the initial marking (or process) m_o themselves.

The occurrence rule is defined as an extension of the elementary object net semantics, satisfying the following additional requirememts:

- Object net processes of the same object net can be identified, e.g. by an identifier.
- In the construction of the LUB only processes from the same object net are considered.
- Object/object interaction is defined as system/object interaction with the additional condition that the interacting object processes are in the same place.

To illustrate the behaviour of object-elementary object systems a cop-process of the fire extinction example of section 2 is constructed. To fit the formal requirements discussed in this paper the object system model for this example is chosen as simple as possible:

- Since a process of the system net is needed, SN is modelled by a contact-free P/T-net with arc-weights restricted to one. Hence, the first net of fig. 2 can be used.
- Since processes of the object nets are needed, the object nets are modelled as Elementary Net Systems (without side conditions). Therefore the nets N_{fi} of fig. 2 are replaced by similar nets of the type in fig. 20.
- The buckets are considered later.

Following these assumptions a (sub-structure of a) cop-process is given in fig. 21. The (sequential) processes labelled "route of fireman i" form a process of the system net SN. They correspond to the four tokens in SN. As known from processes for P/T-nets a state (cut) where two tokens are in the same place is constructed by two places labelled with the name of the place. The places labelled "a" in the processes "route of fireman 1" and "route of fireman 2" give an example.

The (sequential) processes labelled " fireman i" form processes of the object nets N_{fi}. Interaction of a process "route of fireman i" with the process "fireman i" is represented by arcs (sometimes labelled x1, x2, a1, a2, ..). In the same way object/object interactions are represented and labelled by "rvi".

It is possible to construct a mapping Φ associating to each place p of an object net process of N_{fi} a set of places $\Phi(p)$ in the processes "route of fireman i", indicating the "geographical" places where the object net N_{fi} may be situated, assuming it has reached the "state" p. For instance, "fireman 1" is in the place "water" in its states q1" and p1, in the place u in its state p1', and so on. Using this mapping it can be made visible that object/object interactions occur only when the involved object shares the same place, e.g. "fireman 1" and "fireman 2" interact by "rv1" in the shared place "a".

In a straightforward way, also a process of "bucket 1" can be added, as shown in fig. 21. Let Φ_1 be the mapping corresponding to Φ. Then $\Phi_1(\Phi(\text{empty}))$ is the set of all places

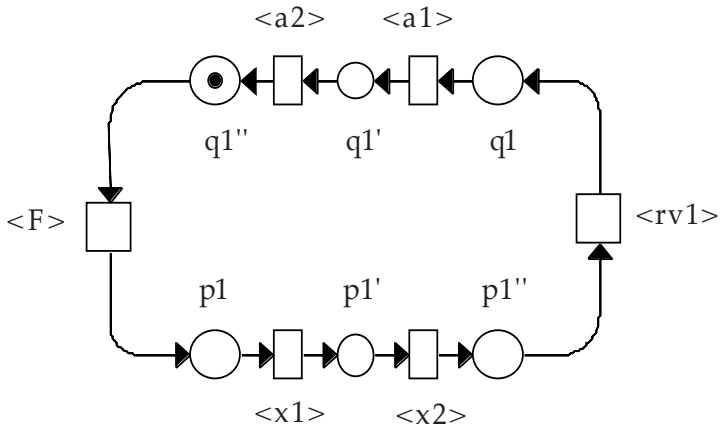


Fig. 20: EN system of fireman 1

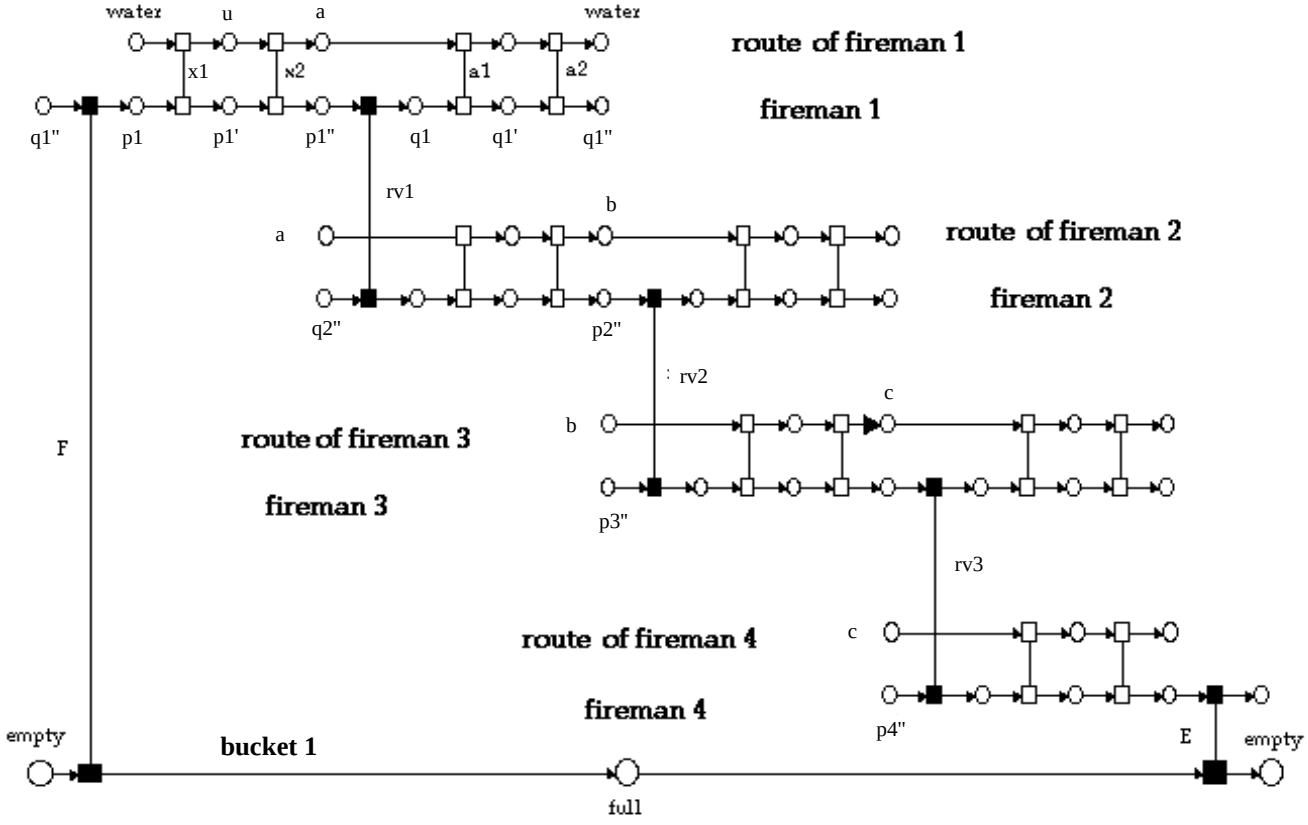


Fig: 21: A Cop-process of the fire extinction example

in the processes "route of fireman i" where "bucket1" may be in its state "empty".

The motivation for discussing this example was to show

- how cop-processes can be constructed for object-elementary object nets,
- that they provide a compact representation of a complex behaviour,
- how several levels of object modelling are included, and
- additional attributes can be made visible.

6. Conclusions

Though the concept "nets as tokens in a net" seemed to be too ambitious for a clear and well-structured approach, this paper shows that by carefully restricting to suitable subclasses the descriptional complexity is manageable. Moreover it was shown that graphical representations are helpful.

By the theorem of this paper a more compact representation of object system processes is presented in the form of cooperating net processes. This representation also formalizes some fundamental aspects of distributed processes, namely by the notion of process morphism.

There is left a lot of open questions from the theoretical point of view and the challenge to incorporate the presented methods in applications.

Appendix: Standard Definitions

This appendix contains some standard definitions of Elementary Net Systems (EN systems) and their processes.

An *EN system* N = (B,E,F,C) [Thiagarajan 87] 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 $C_1 \rightarrow_w C_2$ and $C_1 \rightarrow_w$, respectively. N is called a *state machine* if each place $b \in B$ has at most one input transition ($|{}^\bullet b| \leq 1$) and at most one output transition ($|b^\bullet| \leq 1$) and C contains exactly one token ($|C|=1$).

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

The nonsequential 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 given by a node-labeled causal net $N = (B, E, F, \phi)$ such that $\phi : B \cup E \rightarrow B \cup E$ satisfies

a) $\phi(B) \subset B$ and $\phi(E) \subset E$ and $(\phi(x),\phi(y)) \in F$ for all $(x,y) \in F$

b) $\forall b \in B : {}^\bullet b = \emptyset \Leftrightarrow \phi(b) \in C$

c) ϕ is injective on every B -cut of N

The *initial process* of N_π $init(N_\pi)=(B_C,E_C,F_C,\phi_C)$ consists just of a copy of C, i.e. $B_C=\{b_C \mid b \in C\}$, $E_C = F_C = \emptyset$ and $\phi_C(b_C) = b$. The set of places $B_1 := \{b \in B_\pi \mid b^\bullet = \emptyset\}$ is called a *terminal cut* if all transitions $e \in E_\pi$ have an output place: $e^\bullet \neq \emptyset$. Only finite processes having an initial process and a terminal cut 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 *subprocess generated by* A.
The process $past_{N_\pi}(A) = (B' ,E' ,F' ,\phi')$ is defined by all transitions "before or in A", i.e. $E'_\pi := \{t \in E_\pi \mid t \in A \vee \exists t_1 \in A: t <_{N_\pi} t_1\}$ and all input or output places of E'_π including the initial process, i.e.: $B' := init(N_\pi) \cup \{p \in B \mid \exists t \in E'_\pi : p \in {}^\bullet t \cup t^\bullet\}$. ϕ' is the restriction of ϕ to $B' \cup E'_\pi$. Note that $A=\emptyset$ implies $B' = init(N_\pi)$ and $E'_\pi = \emptyset$.

Given two processes $N_{\pi 1}$ and $N_{\pi 2}$ then a *process morphism* from $N_{\pi 1}$ to $N_{\pi 2}$ is a mapping $\alpha : B_{\pi 1} \cup E_{\pi 1} \rightarrow B_{\pi 2} \cup E_{\pi 2}$ such that $\alpha(B_{\pi 1}) \subset B_{\pi 2}$, $\alpha(E_{\pi 1}) \subset E_{\pi 2}$ and $\forall x,y \in B_{\pi 1} \cup E_{\pi 2}: x <_{N_{\pi 1}} y \Rightarrow \alpha(x) <_{N_{\pi 2}} \alpha(y)$. A restriction of α to transitions $E_{\pi 1}$ and $E_{\pi 2}$ is called a *T-morphism*.

Every firing sequence $w= e_1...e_n \in FS(N)$ uniquely determinates a process $proc(w) = (B ,E ,F ,\phi)$ such that $i < j \Rightarrow \phi(e_i) <_{proc(w)} \phi(e_j)$. 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 $proc_1$ e := $proc_2$ (i.e. $proc_1$ e is the prolongation of $proc_1$ by the transition e and an output place of e).

If for two processes $proc_i$ ($i \in \{1,2\}$) there is a process $proc_3$ such that $proc_i \leq proc_3$, 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 $Q \subset PROC(N)$ and denoted by lub(Q).

7. References

[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.

[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

[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

[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 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