Petri Nets as Token Objects
An Introduction to Elementary Object Nets

Rüdiger Valk

Universität Hamburg, Fachbereich Informatik
valk@informatik.uni-hamburg.de

Abstract. The model of *Elementary Object System* is introduced and motivated by several examples and applications. Object systems support a modeling of systems by Petri nets following the paradigm of Object-Oriented Modeling. They are composed of a *System Net* and one or more *Object Nets* which can be seen as token objects of the system net. By this approach an interesting and challenging two-level system modeling technique is introduced. Similar to the object-oriented approach, complex systems are modeled close to their real appearance in a natural way to promote clear and reliable concepts. Applications in fields like workflow, flexible manufacturing or agent-oriented approaches (mobile agents and/or intelligent agents as in AI research) are feasible. This paper gives an introduction with several examples, but only few definitions and no theorems, which can be found, however, in a more elaborated paper [19].

1 Introduction

1.1 Background

Object-oriented modeling means that software is designed as the interaction of discrete objects, incorporating both data structure and behavior [11]. The notion of *object-oriented modeling* may be understood in (at least) three, somehow different, ways:

– as a *programming style* which is strongly influenced by features and structures of object-oriented programming languages
– as a *modeling concept* leading to system structures that can be easily implemented by object-oriented programming languages
– as a *general modeling principle* producing system models that can be implemented in any language but are in the spirit of the object-oriented paradigm.

This paper intends to contribute to the foundations of object-oriented modeling, in particular with respect to the third of these items within the framework of basic Petri net models. Comparing statements with the goals and advantages of object-oriented modeling on the one hand and Petri net modeling on the other, similar and sometimes identical assertions are found:

- software development by abstraction of objects

- building a language independent design
- better understanding of requirements
- clearer design
- more maintainable systems.

Objects in an object-oriented environment have a dynamical (external) behavior with respect to the basis system and an (internal) behavior, as they change their internal state when interacting with other objects or when being subject of system transactions.

Hence, from a Petri net point of view objects are nets which are token objects in a general system Petri net. We therefore distinguish *Object Nets* from *System Nets*. This paper gives an introduction to some very elementary properties of *Object Systems* composed of a system net and one or more object nets. To keep the model as close as possible to traditional Petri net theory we assume that both system net and object nets are instances of Elementary Net Systems. Therefore this model is called *Elementary Object System (EOS)*. We are not, however, concerned with high level properties of object-oriented modeling and languages, like dynamic instantiation, dynamic binding, inheritance and ploymorphism.

This is in contrast to other approaches within the framework of high level Petri nets ([2], [6], [7], [12]), which introduce object oriented concepts into the Petri net formalism. Our approach has its origins in a work describing the execution of task systems in systems of functional units ([4], [14]). In [16] 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. Further results can be found in [17], [18]. Most results mentioned in this paper are formally elaborated in [19], however, some additional examples are added here.

In the following section we give some examples that will later be used to illustrate the formalism of Elementary Object Systems.

1.2 Examples

Example 1. In the first example task execution by a set of machines is modeled: an object in a production line has to be processed, first by some machine M_1 and then afterwards by machines M_2 or M_3. As it is very natural in the context of manufacturing systems, the process is then reproduced. Besides the machines, operators for the machines are a second type of limited resources: operator O_1 can be working on M_1 or M_2, but not on both at the same time. The same holds for O_2 with respect to M_1 and M_3.

Figure 1 describes this configuration in an intuitive way. Also two of many possible task systems are given. Task system A is composed of four subtasks a_1, a_2, a_3 and a_4 to be sequentially executed on machines M_1, M_2, M_1 and M_3, respectively.

We take an "object-oriented" approach in the sense that the task system is to be modeled as an object that enters machine M_1 and leaves it after execution to be transferred to machine M_2. Attached with the object there is an "execution

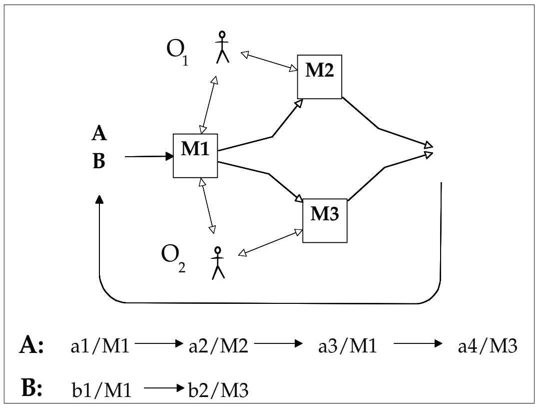

Fig. 1. The example of three machines

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.

Figure 2 gives a Petri net for the machine configuration, which has been used earlier in [8], [9] and [4]. Note that the execution of M_1 is modeled by transitions t_1, t_2 and t_3, t_4 if M_1 is worked by operator O_1 and O_2, respectively. Inscriptions in sharp brackets $<, >$ should be ignored for the moment.

The net is an Elementary Net System (Condition/Event System), with the exception of the objects A and B in place p_1. These are the task systems, as specified in Figure 1. They are represented as Elementary Net System A and B in Figure 3. In the formalism of object nets to be presented here, these nets A and B are considered as token objects. When the subtask a_1 is executed by machine M_1 in a follower marking, net A should be removed from p_1 and appear in the form of A' in place p_6. Hence, both of the following actions are modeled: the task is moved together with its task description *and* the "status" of execution is updated.

Example 2. In the second example we refer to [3], a paper showing a modeling technique for the control of flexible manufacturing systems (*FMS's*) using Petri nets.

In the central example of this paper the manufacturing cell of Figure 4 is studied: "The cell is composed of four machines, *M1*, *M2*, *M3*, and *M4* (each can process two products at a time) and three robots *R1*, *R2*, and *R3* (each one can hold a product at a time). There are three loading buffers (named *I1*, *I2*, *I3*) and three unloading buffers (named *O1*, *O2*, *O3*) for loading and unloading the cell. The action area for robot *R1* is *I1*, *O3*, *M1*, *M3*, for robot *R2* is *I2*, *O2*, *M1*, *M2*, *M3*, *M4* and for robot *R3* is *M2*, *M4*, *I3*, *O1*."

A corresponding P/T-net is shown in Figure 5. When robot *R1* is working, the place *pi_R1* is marked. The capacity restriction for this place is denoted by */1*. This can be seen as a shorthand for an explicit modeling using a complementary place (as done in [3]). The same notation holds for places *pi_R2* and

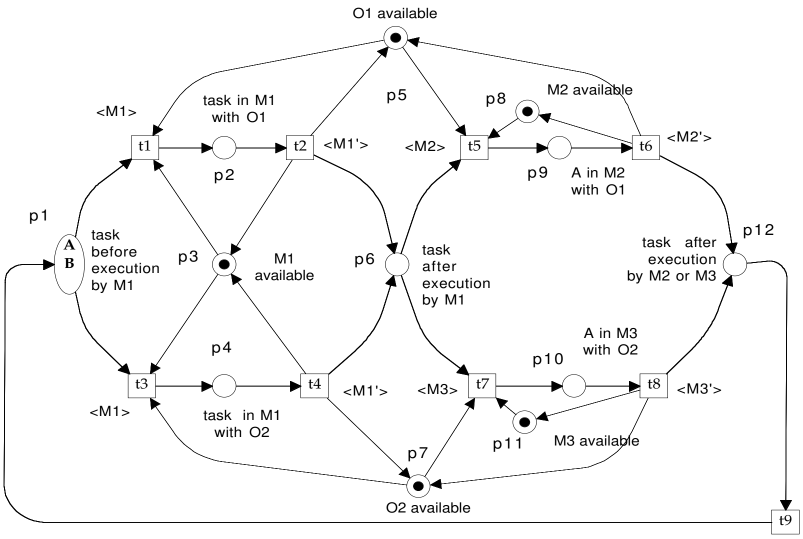

Fig. 2. The three machines' net

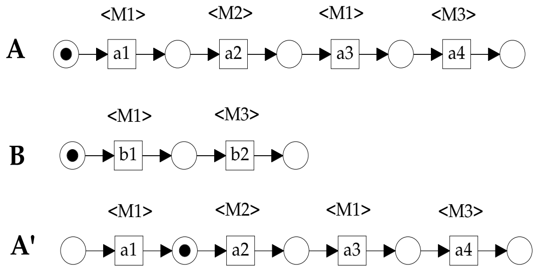

Fig. 3. Task system nets A (A') and B

pi_R3. A transport action from the input buffer *I1* to robot *R1* is denoted by $< I1 \rightarrow R1 >$ etc. The capacity of *2* for the machines is explicitly modeled by complementary places. With these explanations the semantics of the net should be clear; for more details, please refer to [3].

Of particular importance in our context is the observation that usually, in a FMS, different types of parts must be processed.

We cite from [3]:

> The type of part defines which operations must be made on the raw material to get a final product. The type of part is defined by means of its *process plan*. For a given architecture, each process plan is defined by three components (G, I, O), where
>
> 1. G is a (connected) acyclic graph with dummy root: the *operation graph*. Each path from the root to a leaf represents a possible sequence of operations to be performed on the part. The dummy node represents the raw state of a part. A node n of this graph $(n \neq root)$ will be labeled with pairs (n_r, n_0). n_0 stands for the operation to be made on a part of this type, while n_r represents a resource where the operation must be done.
> 2. I refers to the sites from which the parts of the corresponding type can be introduced into the system.
> 3. O refers to the sites from which the parts of the corresponding type can be unloaded from the system.
>
> Figure 6 represents (in the upper part) three such process plans and the operation graph $G1$ of $W1$. The type of product characterizes the process to be made in the cell as follows: 1) a raw product of type $W1$ is taken from $I1$ and, once it has been manufactured, it is moved to $O1$.

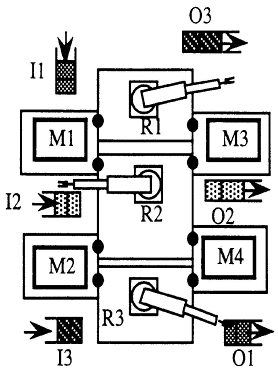

Fig. 4. A flexible manufacturing cell

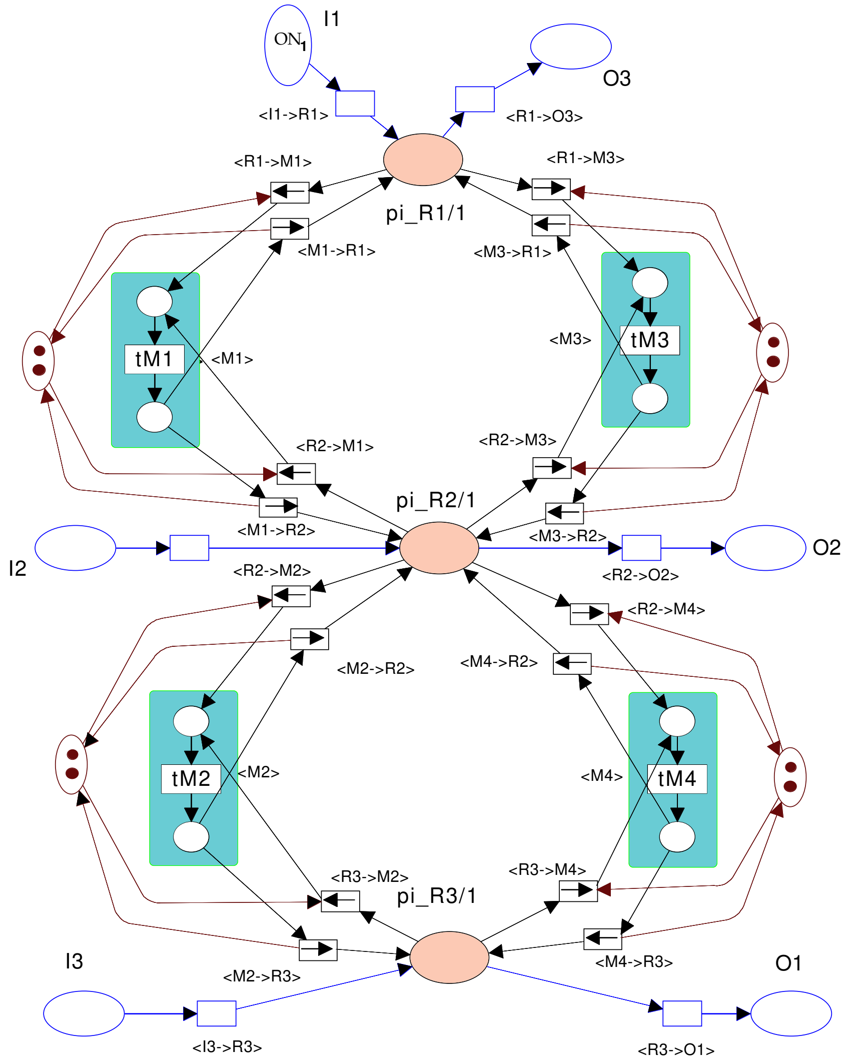

Fig. 5. System net for the FMS cell

In the lower part of Figure 6 an EN system is given, which essentially contains the the same information as $W1$ (by omitting the operations). After the definition of elementary object systems we will use this net as an object net for the example.

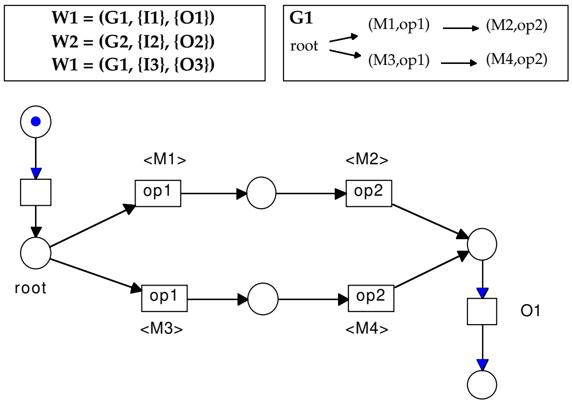

Fig. 6. Task system for the FMS

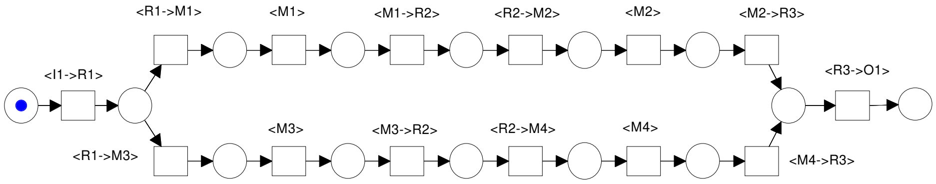

Fig. 7. Refined task system for the FMS

Example 3. In the third example a workflow of the Dutch Justice Department is modeled. It has been used for demonstration of modeling and analysis of workflow applications using Petri nets [1].

The example is introduced in [1] as follows. When a criminal offense has 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, an official examines the case and a prosecutor determines whether the suspect is summoned, charged or that the case is suspended.

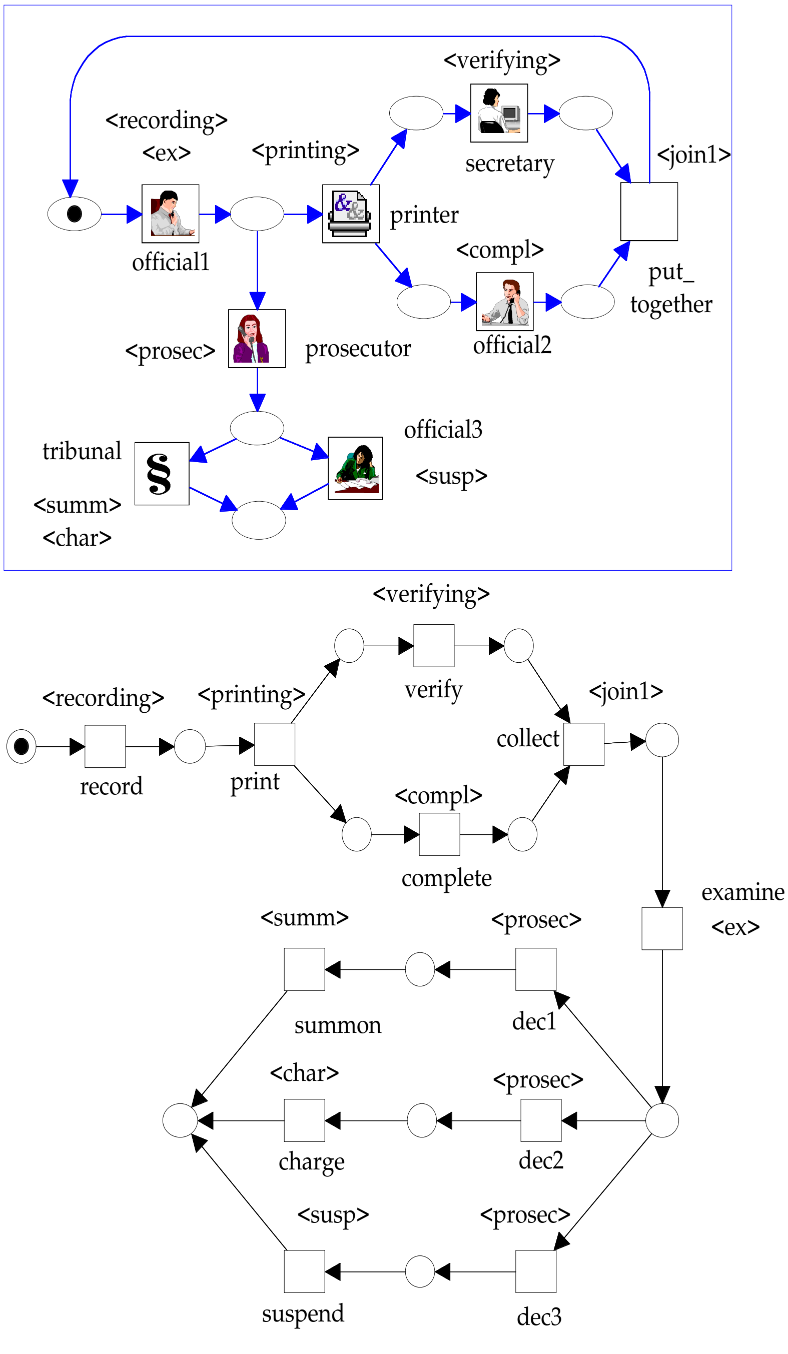

Fig. 8. The work flow example

Originally the case was modeled by a single and "flat" net for the workflow. A slightly modified version is given in the lower part of Figure 8. Observe that verification and completion are concurrent subtasks. The labels in sharp brackets refer to the corresponding functional units (top of Figure 8) executing these subtasks. For instance, "printing" is executed by a printer and "verifying" is executed by the secretary. *Official1* can execute 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 modeled by three transitions *dec1*, *dec2* and *dec3*.

Though being more complex than ordinary workflows (where system nets are not considered), the advantage of this kind of modeling lies in the direct representation of functional units. The system net reflects the organizational structure of the system while the object net 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 such different executions can be used to determine bottlenecks and execution times.

1.3 Overview

In section 2 *Unary Elementary Object Systems* are introduced. Possible system/object interactions are represented by a relation ρ. The model allows for only one object net which may exist, however, in multiple copies. These copies model a behavior of concurrent execution in a distributed system. The notion of *bi-marking* is shown to be adequate only in special cases. To model a behavior including "fork-" and "join-"control structures, the more general notion of *process-marking* is introduced, which is based on the notion of Petri net processes. The corresponding occurrence rule is discussed and the examples, given in section 1.2, are related to the formal definitions. In section 3.1 elementary object systems are introduced in order to model systems with different object nets. Communication between objects is described in the same way as system/object interaction. For formal reasons a different object/object interaction relation σ is introduced. An EOS is called *separated* if the corresponding graphs of ρ and σ are disjoint. To simplify the formalism the occurrence rule is introduced for *simple* EOS only. By this multiple copies of the same object system are avoided. Using a type classification scheme, a subsystem with respect to a particular object net ON_i (the i-*component*) is defined. A special component (the 0-*component*) is reserved for the object class of indistinguishable tokens. As usual, such tokens are used for synchronization and modeling of resources. For illustration of the model a distributed and object-oriented version of the *five philosophers model* is given.

2 Object Systems

2.1 Unary EOS and Bi-Markings

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 [13]. 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 $C_1 \rightarrow_t C_2$. If t is enabled in C_1 we write $C_1 \rightarrow_t$. These notions are extended to words $w \in E^*$, as usual, and written as $C_1 \rightarrow_w C_2$. 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 ($|C| = 1$). $FS(N) := \{w \in E^* | C \rightarrow_w\}$ is the set of *firing* or *occurrence sequences* of N, and $R(N) := \{C_1 | \exists w : C \rightarrow_w C_1\}$ is the set of reachable markings (or cases), also called the *reachability set* of N (cf. [10]). We will also use *processes* of EN systems in their standard definition [10].

Definition 4. *A* unary elementary object system *is a tuple* $EOS = (SN, ON, \rho)$ *where*

- $SN = (P, T, W, \mathbf{M}_0)$ *is an EN system with* $|\mathbf{M}_0| = 1$, *called* system net *of* EOS,
- $ON = (B, E, F, \mathbf{m}_0)$ *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.*

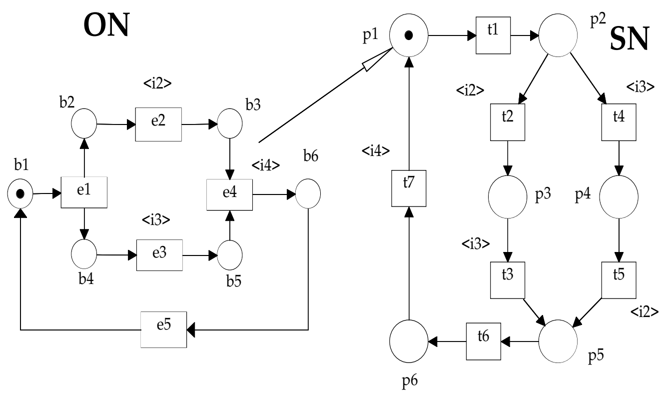

Fig. 9. Elementary object system “ser-task”

Figure 9 gives an example of an elementary object system with the components of an object net ON on the left and a system net SN on the right. The interaction relation ρ is given by labels $< i_n >$ at t and e iff $t \rho e$ (“i_n” stands for interaction number n, which has no other meaning apart from specifying interacting transitions).

Before proceeding to the formalization we describe the intuition behind the occurrence rule to be defined later. The object net ON of Figure 9 should be thought of lying in place p_1 of the system net SN. It is represented by a token in that place. Since there is no label at transition t_1 the object net ON is moved to p_2 by the occurrence of transition t_1. Since it does not change its marking such an occurrence is called a *transport*. In a dual sense also transition e_1 of ON can occur without interacting with the system net. Therefore such an occurrence is called *autonomous*. Now, both nets ON and SN have reached a marking where e_2 and t_2 are activated (as well as e_3 and t_4), when considered as separated EN systems. Since they bear the same label ($< i_2 >$ in this case) they must occur simultaneously in the object system.

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 $t\rho := \{e \in E | (t, e) \in \rho\}$ and $\rho e := \{t \in T | (t, e) \in \rho\}$. Then $t\rho = \emptyset$ means that there is no element in the interaction relation with t.

Definition 5. *A* bi-marking *of an unary elementary object system* $EOS = (SN, ON, \rho)$ *is a pair* $(\mathbf{M}, \mathbf{m})$ *where* $\mathbf{M}$ *is a marking of the system net* SN *and* $\mathbf{m}$ *is a marking of the object net* ON.

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

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

c) A transition $e \in E$ *is* activated *in a bi-marking* $(\mathbf{M}, \mathbf{m})$ *of a* EOS *if* $\rho e = \emptyset$ *and* e *is activated in* $\mathbf{m}$ *. Then the follower bi-marking* $(\mathbf{M}', \mathbf{m}')$ *is defined by* $\mathbf{m} \rightarrow_e \mathbf{m}'$ *(w.r.t. ON) and* $\mathbf{M}' = \mathbf{M}$. *We write* $(\mathbf{M}, \mathbf{m}) \rightarrow_{[\lambda,e]} (\mathbf{M}', \mathbf{m}')$ *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 9, 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.

The unary EOS "con-task" of Figure 10 has the same object net as "ser-task" of Figure 9 (with the exception of the new label $< i_1 >$), but a different system net. By transition t_1 the object net is duplicated. After this event task execution is concurrently performed on two instances of the same object net. A possible occurrence sequence is:

$$[t_1, e_1], [t_3, e_3], [t_2, e_2], [t_7, e_4], [\lambda, e_5], [t_8, \lambda].$$

The bi-marking reached after the first three steps is $(\{p_3, p_5\}, \{b_3, b_5\})$, which activates the pair of transitions $[t_7, e_4]$.

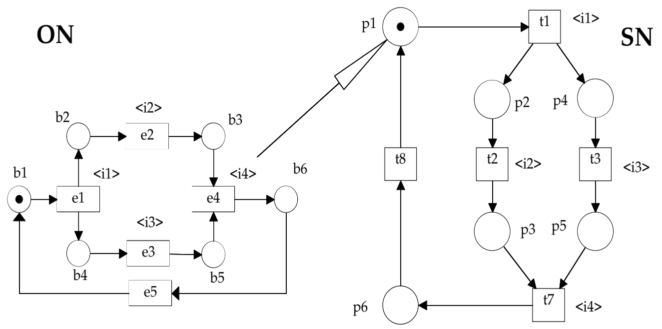

Fig. 10. Elementary object system "con-task"

2.2 Counter Examples

The given occurrence sequence of the EOS "con-task" correctly reflects the intended behavior: subtasks e_2 and e_3 are concurrently executed and the "outcome" of these executions is collected by the "join"-transition t_7. Using bi-markings and the corresponding occurrence sequence semantics may however result in a counter-intuitive behavior. For the EOS "counter1" in Figure 11 the occurrence sequence

$$[t_1, e_1], [t_3, e_3], [t_2, e_6]$$

leads to the bi-marking $(\{p_3, p_5\}, \{b_5, b_6\})$, which activates $[t_7, e_7]$. This somehow "strange" behavior is also due to the fact that not a really distributed

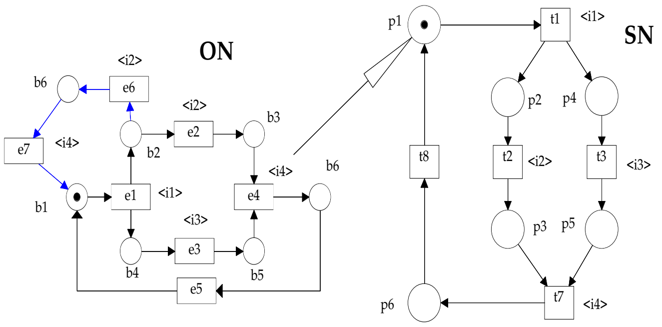

Fig. 11. Elementary object system "counter1"

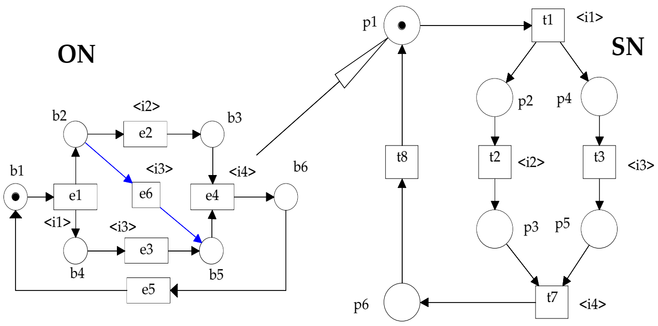

Fig. 12. Elementary object system "counter2"

system is generated. Although transition t_1 should create two independent instances of ON only one instance is referenced. A different choice would be to associate to each instance of ON an individual "local" marking. For the EOS "con-task" the marking activating the pair of transitions $[t_7, e_4]$ would have the form: $((ON, \{b_3, b_4\}); (ON', \{b_2, b_5\}))$, where ON and ON' are copies lying in the places p_3 and p_5. We will refer to markings of this form as *object markings.*

But also this choice of a marking definition is not satisfying. In the EOS "counter2" of Figure 12 the marking $((ON, \{b_3, b_4\}); (ON', \{b_4, b_5\}))$ would be reachable by the occurrence sequence

$$[t_1, e_1], [t_2, e_2], [t_3, e_6].$$

It is obvious that in this case an activation of the pair of transitions $[t_7, e_4]$ is not adequate since the instances in the input places of t_7 result from conflicting executions of the same branch of ON. It is therefore not a suitable formalization of a well-formed "fork/join" control structure.

2.3 Process Markings

As introduced and formalized in [18], [19], a solution to the problems addressed in the previous section is possible by using *Process markings (P-markings)* instead of bi-markings. For a unary EOS, where the referenced object net is unique (modulo the current marking), a P-marking associates to every place of the system net a process of the object net. Processes are represented by causal nets in their standard definition for EN systems (see [10], [19]).

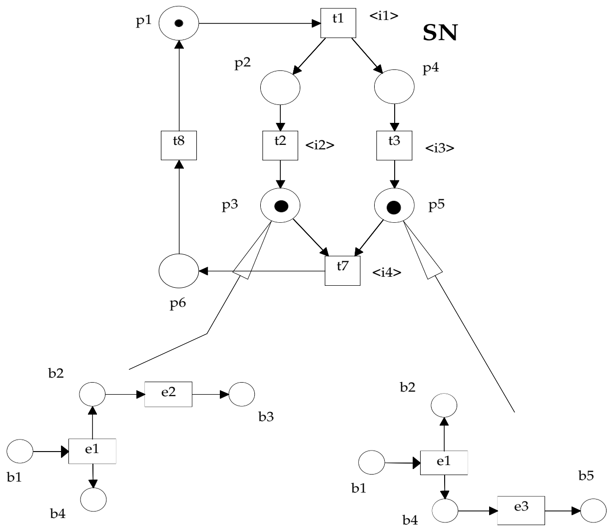

Fig. 13. Elementary object system "con-task" with P-marking

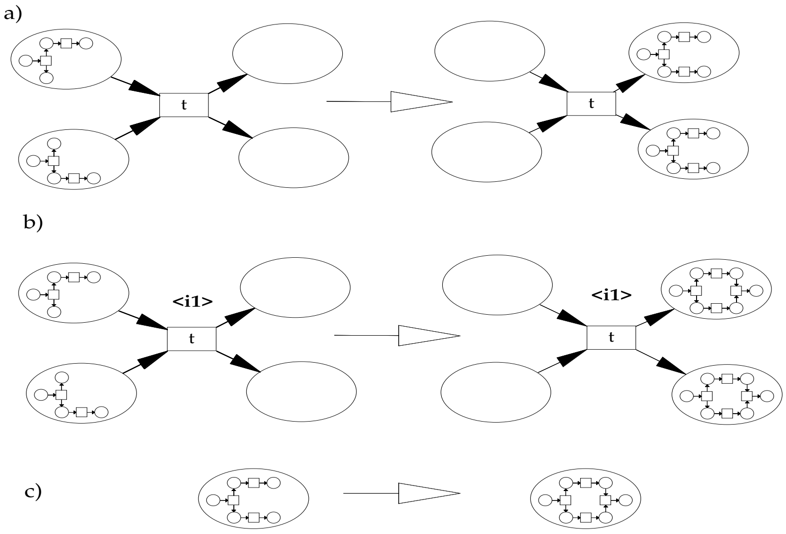

Fig. 14. P-marking occurrence rule

To give an example, in Figure 13 a P-marking for "con-task" is given, corresponding to the marking activating $[t_7, e_4]$, which has been discussed in section 2.2. It shows the (partial) processes of concurrent task execution in the input places of transition t_7. Different to bi-markings, the history of the partial execution is recorded, which allows for a more adequate detection of "fork/join-structures".

Informally the conditions for activation of a transition and the definition of a follower P-marking is described in the following. The cases a), b) and c) are represented graphically in Figure 14. A transition e of an EN system is called *activated* in a process *proc* if the process can be enlarged by this event. The new process is unique and denoted by $proc_e := proc \circ e$. Generally, a process $proc_1$ can be called *smaller* than $proc_2$ if $proc_1$ is an "initial part" of $proc_2$. With respect to this partial ordering on the set of all processes of an EN system, for a subset of such processes a *least upper bound (lub)* may exist. It is constructed by "combining" all the processes in a consistent way.

a) *Transport*: $t \in T$, $t\rho = \emptyset$
   1. Each input place $p_i \in {}^\bullet t$ contains a process $proc_i$ of ON.
   2. The set $\{proc_i | p_i \in {}^\bullet t\}$ of these processes has a least upper bound $proc_{lub}$.
   3. $[t, \lambda]$ is *activated* if conditions 1. and 2. hold.
   4. The follower P-marking is obtained by removing all processes from the input places of t and by adding the process $proc_{lub}$ to all output places (recall that there are no side conditions in standard EN systems).

b) *Interaction*: $t \in T$, $e \in E$, $(t, e) \in \rho$
   1. Each input place $p_i \in {}^\bullet t$ contains a process $proc_i$ of ON.

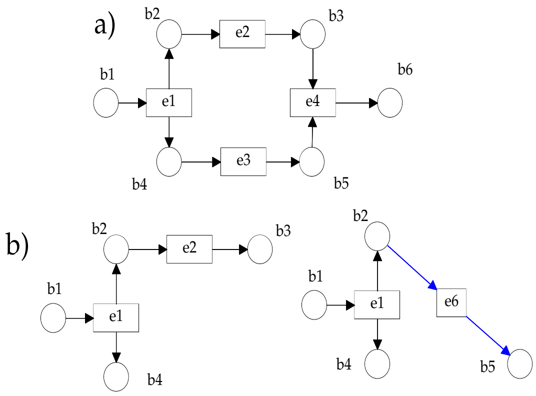

Fig. 15. a) Follower P-marking, b) P-marking for "counter2"

2. The set $\{proc_i|p_i \in {}^\bullet t\}$ of these processes has a least upper bound $proc_{lub}$.
3. e is activated in $proc_{lub}$ i.e. $proc_e := proc_{lub} \circ e$ is defined.
4. $[t, e]$ is *activated* if conditions 1., 2. and 3. hold.
5. The follower P-marking is obtained by removing all processes from the input places of t and by adding the process $proc_e$ to all output places.

c) *Object-autonomous event*: $proc$ in p, $e \in E$, $\rho e = \emptyset$
1. Transition e is activated for $proc$ i.e. $proc_e := proc \circ e$ is defined.
2. $[\lambda, e]$ is *activated* if condition 1. holds.
3. The follower P-marking is obtained by substituting $proc_e$ for $proc$ in the place p.

According to case b) the pair $[t_7, e_4]$ is activated in the P-marking of Figure 13. The follower P-marking is given by the process of Figure 15 a) in the place p_6. A P-marking as in Figure 13 but for the EOS "counter2" is given in Figure 15 b). Now, despite the fact that both input conditions b_3 and b_5 of the ON-transition e_4 are holding, the pair $[t_7, e_4]$ is not activated, since there is no least upper bound for the two processes.

We conclude that bi-markings are sufficient in many cases but not in general. Since they are much simpler to formalize and implement they should be used whenever possible. Notice that P-marking semantics is similar to bi-marking semantics for the example of "counter1", but not for "counter2". The EOS "counter1" was useful to show some counter-intuitive behavior. This is a result of the following property of the EOS "counter1". While the system net transition t_7 requires input from two input places ("input channels of a distributed system") only one of them is effectively used, namely the object system instance from input place p_3. Such a behavior could be excluded by the P-marking semantics by requiring in case b) of the occurrence rule given in this section that process $proc_i$ of each input place p_i contains a precondition of the object net transition $e \in E$ that is indispensable for the activation of e. But this is out of the scope of the present paper. Also for reasons of simplicity, not due to any fundamental problems, in the next chapter we introduce *simple* elementary object systems, where multiple instances of object nets are excluded. The model is extended, however, by allowing more than one *different* object nets that may communicate. Summing up the discussion:

- Bi-markings are references to the marking of a single object net. Different "copies" are nothing but references to the same object. They are preferable due to their simple structure, but have their limit in a distributed environment.
- Object-markings represent copies of objects and not only references. They do not reflect "fork/join"-control structures correctly. This is due to the existence of "superflues" markings in the copies.
- P-markings also represent copies of objects and partially record the past of computations, allowing to merge distributed computations consistently.

There are, however, a lot of formal reasons to prefer P-marking semantics. As shown in [19] a suitable process notion for elementary object systems can be formalized. A theorem is given there, characterizing such a process by a triple $(proc_1, proc_2, \varphi)$, where $proc_1$ and $proc_2$ are processes of the EN systems SN and ON, respectively (in the standard notion of [10]), and φ is a *process morphism* (i.e. a net morphism between causal nets) having particular properties. This theorem strongly relates the theory of object nets to the traditional Petri net theory and therefore proves the compatibility of the concepts.

2.4 Examples

After having introduced unary elementary object systems by formal definitions we take another look at the examples of section 1.2.

Example 1 is modeled by the EN systems in the Figures 2 and 3. If we delete all places containing an indistinguishable token (like p_3) in Figure 2 and replace the letters A and B by a single indistinguishable token, we obtain a system net SN, that - together with the object net A from Figure 3 - represents a unary EOS according to definition 4. The labels in sharp brackets (like $< M1 >$) define the interaction relation ρ. The restricted model does not represent the resources. If these are to be included the EOS is to be interpreted as simple EOS according to definition 8, below. Then the EN systems A and B are interpreted as object nets $ON_1 = A$ and $ON_2 = B$. The object/object interaction relation σ is empty and the arc type function must be defined appropriately (e.g. $type(p_1, t_1) = \{1, 2\}$)

Example 2 has a representation by a simple EOS with the system net SN in Figure 5 and the object net ON_1 in Figure 6. Additional object nets and appropriate arc type functions can easily be added. A closer look, however, shows that the model does not work correctly. This is due to the conflicting granularity of interacting transitions. To solve the problem, all labels different from $< M1 >$, $< M2 >$, $< M3 >$and $< M4 >$ could be removed from SN. By this deletion the corresponding interacting transitions are transformed into transports. Alternatively, the granularity of the object net could be increased by adding some interacting transitions, as shown in Figure 7. The object net ON plays the role of a process plan as defined in [3] with the additional information on the current state (i.e. the marking of ON).

Example 3 is modeled by the system net SN and the object net ON of Figure 8. It is a unary EOS with concurrent objects where the bi-marking semantics is sufficient. The object net can be seen as a document that can be printed in multiple copies. It contains information how to proceed by the administration and on the current state of this process. Two copies can differ only in the current state (marking).

3 Communicating Objects

3.1 Definitions

In this section unary elementary object systems are extended in such a way that different object nets move through 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 6. *An* elementary object system *is a tuple* $EOS = (SN, \widehat{ON}, Rho, type, \widehat{\mathbf{M}})$ *where*

- $SN = (P, T, W)$ *is a net (i.e. an EN system without initial marking), called* system net *of EOS,*
- $\widehat{ON} = \{ON_1, ..., ON_n\}$ $(n \geq 1)$ *is a finite set of EN systems, called* object systems *of EOS, denoted by* $ON_i = (B_i, E_i, F_i, \mathbf{m}_{0i})$
- $Rho = (\rho, \sigma)$ *is the* interaction relation, *consisting of a* system/object interaction relation $\rho \subseteq T \times \boldsymbol{E}$ *where* $\boldsymbol{E} := \bigcup\{E_i | 1 \leq i \leq n\}$ *and a symmetric* object/object interaction relation $\sigma \subseteq (\boldsymbol{E} \times \boldsymbol{E}) \setminus id_E$*,*
- $type : W \to 2^{\{1,\dots,n\}} \cup \mathbb{N}$ *is the* arc type function, *and*
- $\widehat{\mathbf{M}}$ *is a marking as defined in definition 7.*

Figure 16 gives a graphical representation of an elementary object system with a system net SN and three object nets ON_i $(1 \leq i \leq 3)$. 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 labeled 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 7. *The set* $\boldsymbol{Obj} := \{(ON_i, \mathbf{m}_i) | 1 \leq i \leq n, \mathbf{m}_i \in R(ON_i)\}$ *is the* set of objects *of the EOS. An* object-marking *(O-marking) is a mapping* $\widehat{\mathbf{M}} : P \to 2^{\mathbf{Obj}} \cup \mathbb{N}$ *such that* $\widehat{\mathbf{M}}(p) \cap \mathbf{Obj} \neq \emptyset \Rightarrow \widehat{\mathbf{M}}(p) \cap \mathbb{N} = \emptyset$ *for all* $p \in P$.

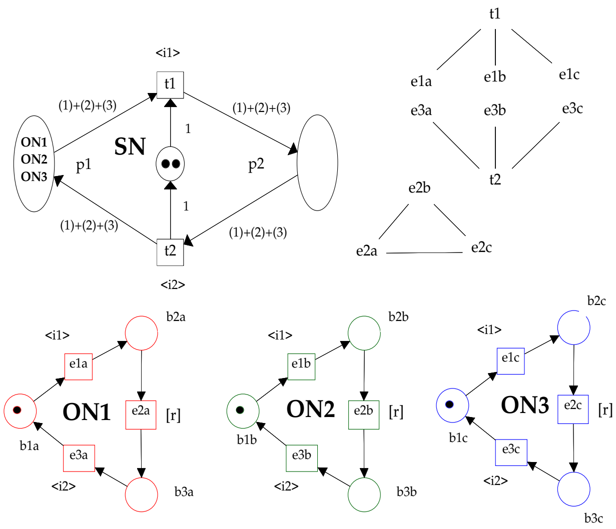

Fig. 16. A simple Elementary object system with 3 objects

A marking of an EOS is a generalization of a bi-marking to more than a single object net. Ordinary tokens easily fit into the concept since they represent a particular object class. The (initial) O-marking of the EOS in Figure 16 is obvious. By restriction to a particular object type from EOS we obtain a unary EOS (i-component, $1 \leq i \leq n$). 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.

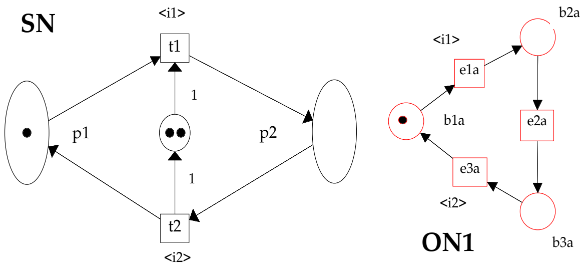

Fig. 17. The 1-component EOS(1) of Figure 16

Definition 8. *Let* $EOS = (SN, \widehat{ON}, Rho, type, \widehat{\mathbf{M}})$ *be an elementary object system as given in definition 6, but in some arbitrary marking* $\widehat{\mathbf{M}}$.

– $Rho = (\rho, \sigma)$ *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* $SN(i) = (P, T, W(i), \mathbf{M}_{0i})$ *defined by* $W(i) = \{(x,y) | i \in type(x,y)\}$ *and* $\mathbf{M}_{0i}(p) = 1$ *iff* $(ON_i, \mathbf{m}_i) \in \widehat{\mathbf{M}}(p)$*. The* 0-component *(zero-component) is the P/T-net* $SN(0) = (P, T, W(0), \mathbf{M}_{00})$ *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* $k \in \widehat{\mathbf{M}}(p)$*.*
- *The subnet* $SN(1..n) = (P, T, W(1..n), M_{1..n})$*, where* $W(1..n) = \bigcup\{W(i) \mid 1 \le i \le n\}$ *and* $M_{1..n}(p) = \widehat{\mathbf{M}}(p) \cap \mathbf{Obj}$ *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 9. For each $i \in \{1, ..., n\}$ the i-component $EOS(i) := (SN(i), ON_i, \rho(i))$ is a unary EOS, where $\rho(i) := \rho \cap (T \times E_i)$.

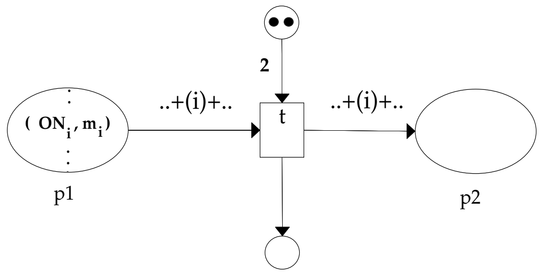

Fig. 18. Occurrence rule for simple EOS

The EOS from Figure 16 is simple since each $SN(i)$ $(1 \le i \le 3)$ 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 Figure 17). Dropping the condition that $SN(1..n)$ is a structural state machine would lead to inconsistencies in the definition of the dynamical behavior (definition 10). 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 behavior 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 interaction 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 definition 5 whereas object/object interactions are associated with case c) of this definition.

Definition 10. *Let* $EOS = (SN, \widehat{ON}, Rho, type, \widehat{\mathbf{M}})$ *be an elementary object system as in definition 6 and* $\widehat{\mathbf{M}} : P \to 2^{\mathbf{Obj}} \cup \mathbb{N}$ *an O-marking (definition 7) 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* $\widehat{\mathbf{M}}$ *(denoted* $\widehat{\mathbf{M}} \to_t$*) if* $t\rho = \emptyset$ *and the following holds:*
   1. t *is activated in the zero-component of* SN *(definition 8) (i.e. in the P/T-net part)*
   2. *By the state machine property there is at most one type* $i \in \{1, \ldots, n\}$ *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* $(ON_i, \mathbf{m}_i) \in \widehat{\mathbf{M}}(p_1)$.*(cf. Figure 18)*

If t *is activated, then* t *may occur* $(\widehat{\mathbf{M}} \to_t \widehat{\mathbf{M}}')$ *and the* follower marking $\widehat{\mathbf{M}}'$ *is defined as follows: with respect to the zero-components tokens are changed according to the ordinary P/T-net occurrence rule. In case of* a2) $(ON_i, \mathbf{m}_i)$ *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* $\widehat{\mathbf{M}}$ *(denoted* $\widehat{\mathbf{M}} \to_{[t,e]}$*) if in addition to case a) transition* e *is also activated for* ON_i *in* $\mathbf{m}_i$. *Instead of* $(ON_i, \mathbf{m}_i)$ *the changed object* $(ON_i, \mathbf{m}_{i+1})$ *where* $\mathbf{m}_i \to_e \mathbf{m}_{i+1}$ *is added.*

c) A pair $[e_i, e_j] \in E_i \times E_j$ *with* $e_i \sigma e_j$ *is* activated *in* $\widehat{\mathbf{M}}$ *(denoted* $\widehat{\mathbf{M}} \to_{[e_i,e_j]}$*) if for some place* $p \in P$ *two objects* $(ON_i, \mathbf{m}_i) \in \widehat{\mathbf{M}}(p)$ *and* $(ON_j, \mathbf{m}_j) \in \widehat{\mathbf{M}}(p)$ *are in the* <u>*same*</u> *place* p *and* $\mathbf{m}_i \to_{e_i} \mathbf{m}_{i+1}$ *and* $\mathbf{m}_j \to_{e_j} \mathbf{m}_{j+1}$. *In the follower marking* $\widehat{\mathbf{M}}'$ *the objects* $(ON_i, \mathbf{m}_i)$ *and* $(ON_j, \mathbf{m}_j)$ *in* p *are replaced by* $(ON_i, \mathbf{m}_{i+1})$ *and* $(ON_j, \mathbf{m}_{j+1})$, *respectively.*

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

3.2 Distributed Philosophers

To apply the definition to a well-known example, we consider the case study of *The hurried Philosophers.* It has been proposed by C. Sibertin-Blanc [12] to test expressive and analytic power of languages merging Petri nets and concepts of the object-oriented approach. We adopt here the distributed character of this extension, but are not concerned with dynamic instantiation, dynamic binding, inheritance and ploymorphism.

Consider the system net SN in Figure 20. There are five object nets $ph_1, \ldots, ph_5$ 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. Two of these object nets, namely ph_i and ph_k are shown in Figures 20 and 19.

In a truly distributed environment the philosophers can only communicate by sending messages. In "his" place p_i philosopher ph_i finds an object net shr_i: *fork*

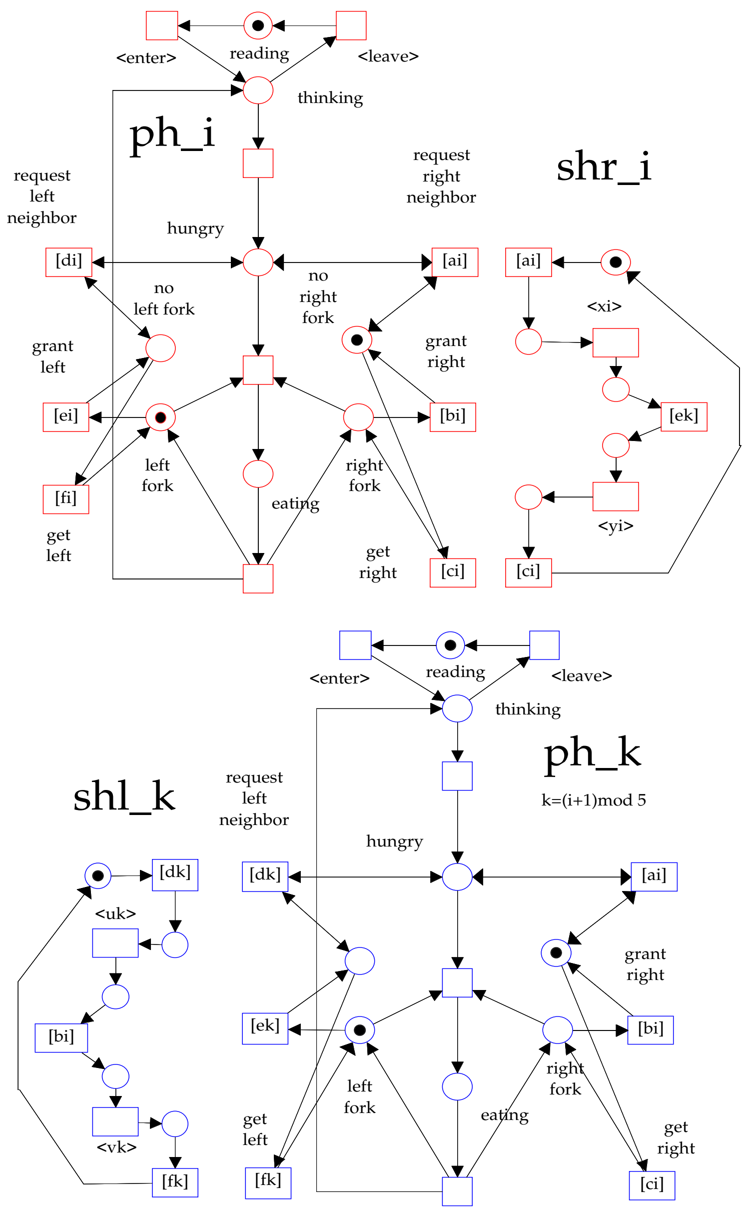

Fig. 19. Five philosophers objects nets

shuttle right, that can be used to send a request to his right neighbor ph_k by the interaction $[a_i]$ (see Figure 19). The shuttle then moves to p_k using interaction $< x_i >$ to take the fork of ph_k using interaction $[e_k]$, 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 $[c_i]$. The type of this object net is (s_i) and the corresponding inscriptions are given on the arcs. In a symmetrical way ph_k uses shuttle shl_k (*fork shuttle left*) to obtain the fork back. Note, that by typed arcs a philosopher ph_i can reach his "place" p_i, but none of the others $p_j, (j \neq i)$, at the table.

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 requesting 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 present neighbor.) Such variants of specifications are out of the scope of this paper.

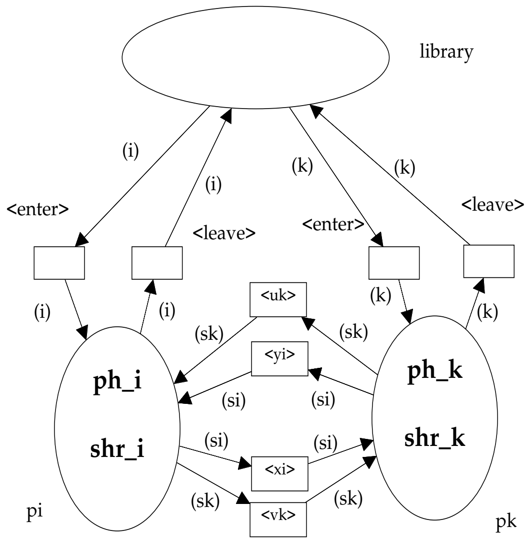

Fig. 20. Five philosophers system net

3.3 Invariants

Since the partners for communication are fixed in this example, by merging communicating transitions, an ordinary net (see [19]) can be constructed, representing the behavior of shuttle exchange. By restriction to only two neighboring philosophers, this net can be seen as a communication protocol for *distributed mutual exclusion*, being similar to the methods of [15] and [5].

It is interesting to compare the different structures of these solutions using P-invariants. While the approach in [15] and [5] reflects a typical *request/grant* scheme, as known in protocol design the object oriented approach presented here contains P-invariants, describing the cyclic behavior of the fork shuttle. By this the difference of *object oriented design* is reflected in the formal structure of of the net graph and the P-invariants. For the proof of properties like mutual exclusion object overlapping P-invariants are need. As a case study has shown, they can be computed from the P-invariants of the individual objects.

4 Conclusion

The increasing importance of the object-oriented modeling paradigm leads to the introduction of object nets. There is, however, a huge number of alternatives for doing so. Up to now no fundamental studies are known as in the case of the basic Petri net model. We introduce such a basic model of object nets using elementary net systems. They are motivated by several examples arising from applications and by the first study of fundamental properties like distributed computations. Unary elementary object nets allow the study of such effects on an elementary level. It is expected that this will give insight to similar properties of high level object nets. Simple elementary object nets include more than one object which may interact. This is illustrated by extending the five philosophers model to a distributed environment.

References

1. W.v.d. Aalst. private communication. 1997.
2. U. Becker and D. Moldt. Object-oriented concepts for coloured petri nets. In *Proc. IEEE Int. Conference on Systems, Man and Cybernetics*, volume 3, pages 279–286, 1993.
3. J. Ezpeleta and J.M. Colom. Automatic synthesis of colored petri nets for the control of fms. *IEEE Transactions on Robotics and Automation*, 13(3):327–337, 1997.
4. E. Jessen and R. Valk. *Rechensysteme – Grundlagen der Modellbildung.* Springer-Verlag, Berlin, 1987.
5. E. Kindler and R. Walter. Message passing mutex. In J. Desel, editor, *Structures in Concurrency Theory*, Berlin, 1995. Springer-Verlag.
6. C.A. Lakos. Object petri nets. Report TR94-3, Computer Science Depart., University of Tasmania, 1994.
7. C.A. Lakos. From coloured petri nets to object petri nets. In M. Diaz G. De Michelis, editor, *Application and Theory of Petri Nets*, number 935 in LNCS, pages 278–297, Berlin, 1995. Springer-Verlag.
8. J.L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall, Englewood Cliffs, 1981.
9. W. Reisig. System design using petri nets. *Informatik Fachberichte*, 74:29–41, 1983.

10. G. Rozenberg. Behaviour of elementary net systems. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Central Models and their Properties*, number 254 in LNCS, pages 60–94. Springer-Verlag, Berlin, 1987.
11. J. Rumbaugh et al. *Object-Oriented Modeling and Design*. Prentice-Hall, London, 1991.
12. C. Sibertin-Blanc. Cooperative nets. In R. Valette, editor, *Application and Theory of Petri Nets*, number 815 in LNCS, pages 471–490, Berlin, 1994. Springer-Verlag.
13. P.S. Thiagarajan. Elementary net system. In W. Brauer, W. Reisig, and G. Rozeberg, editors, *Petri Nets: Central Models and their Properties*, number 254 in LNCS, pages 26–59. Springer-Verlag, Berlin, 1987.
14. R. Valk. Nets in computer organisation. In W. Brauer, W. Reisig, and G. Rozeberg, editors, *Petri Nets: Central Models and their Properties*, volume 255, pages 218–233. Springer-Verlag, Berlin, 1987.
15. R. Valk. On theory and practice: an exercise in fairness. *Petri Net Newsletter*, 26:4–11, 1987.
16. R. Valk. Modelling concurrency by task/flow EN systems. In *3rd Workshop on Concurrency and Compositionality*, number 191 in GMD-Studien, St. Augustin, Bonn, 1991. Gesellschaft für Mathematik und Datenverarbeitung.
17. R. Valk. Petri nets as dynamical objects. In *Workshop Proc. 16th International Conf. on Application and Theory of Petri Nets, Torino, Italy*, June 1995.
18. R. Valk. On processes of object petri nets. Bericht 185/96, Fachbereich Informatik, Universität Hamburg, 1996.
19. R. Valk. Concurrency in communicating object petri nets. In F. DeCindio G.A. Agha, editor, *to appear in: Advances in Petri Nets*, LNCS. Springer-Verlag, Berlin, 1998.