

Object Petri Nets

Using the Nets-within-Nets Paradigm.

Rüdiger Valk.

revised version from

Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets: Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*, pages 819-848. Springer-Verlag, Berlin, Heidelberg, New York, 2004.

Copyright Springer-Verlag Berlin

Object Petri Nets

Using the Nets-within-Nets Paradigm

Rüdiger Valk

Universität Hamburg, Vogt-Kölln-Str.30, D-22527 Hamburg, Germany
valk@informatik.uni-hamburg.de

Abstract. The nets-within-nets paradigm provides an innovative modelling technique by giving tokens themselves the structure of a Petri net. These nets, called *token nets* or *object nets*, also support the object oriented modelling technique as they may represent real world objects with a proper dynamical behaviour. Between object nets and the surrounding net, called *system net*, various interaction mechanisms exist as well as between different object nets. This introduction into the field of object Petri nets starts with small examples and proceeds by giving formal semantics. Some of the examples are modelled within the formalism of the Renew tool. Finally the differences between reference and two kinds of value semantics are discussed.

1 Nets within Nets

Tokens in a Petri net place can be interpreted as objects. In place/transition nets (P/T nets), in most cases these objects represent resources or indicate the state of control. More complex objects are modelled by typed tokens in coloured Petri nets. Object-oriented modelling, however, means that software is designed as the interaction of discrete objects, incorporating both data structure and behaviour [1]. From a Petri net point of view it is quite natural to represent such objects by tokens, that are nets again. We denote this approach as the “nets-within-nets paradigm”.

In many applications objects not only belong to a specific environment but are also able to switch to a different one. Examples of such objects are agents, including the classical meaning of persons belonging to a secret service, as well as software modules in the context of agent-oriented programming. In Fig. 1a) the current environment of agent X is denoted as “location A”, which may be a logical state or a physical site. The arrow represents a possible transition to location B. A structurally similar situation is given with a mobile computer switching between “security environments” A and B (Fig. 1b). In Fig. 2a) the object is a task to be executed on a machine A together with a plan for the execution procedure. As before, they can move to machine B. Finally, in Fig. 2b) a workflow is with an employee A, moving to employee B afterwards. Note, that in all of these examples, also the internal state of the object is changed when

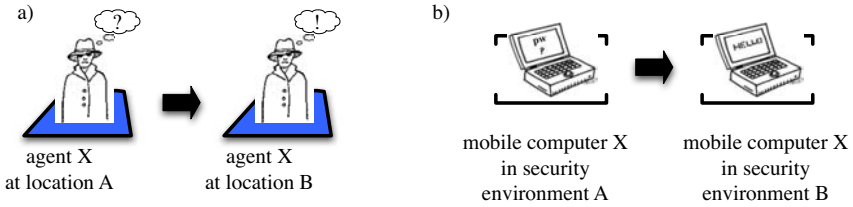


Fig. 1. Moving objects I

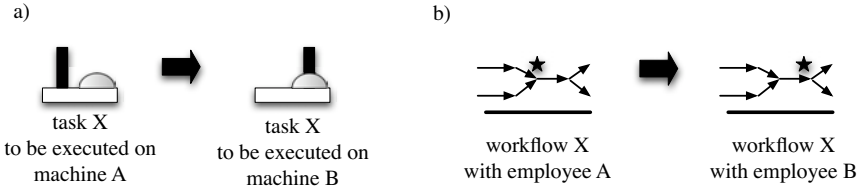


Fig. 2. Moving objects II

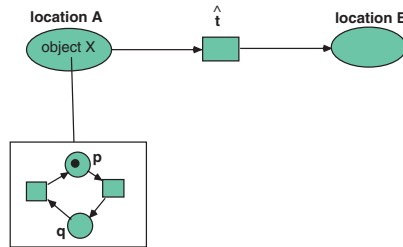


Fig. 3. Object system transition with token net

moving to a different location. Abstracting from these examples, we model the movement or switch of an object X by a transition, which is enabled by the token “object X ”, as shown in Fig. 3. In addition, as the object has a dynamical behaviour, say alternating states p and q , the token is again a marked net. It is therefore called a “token net”. A token net is also called *object net* in distinction to the *system net*, to which it belongs. The whole system is then called an *object net system* or shortly *object system*. In Fig. 4 the movement of the net-token is shown as the firing of transition \hat{t} . Obviously, also the token net can fire autonomously without being moved (Fig. 5). Both, transport and autonomous firing can interleave, but are to be considered as concurrent actions. This should be distinguished from a situation, where these transition occurrences are synchronised, i.e. the object moves if and only if some object net transition occurs. Such an action may be triggered by the object net, by the system net, or by both of them. Therefore such a situation is denoted by the neutral term “interaction”. Interacting transitions are labelled by a corresponding symbol, such as $\langle i \rangle$ in Fig. 6. Finally in this introduction we discuss two different semantics of

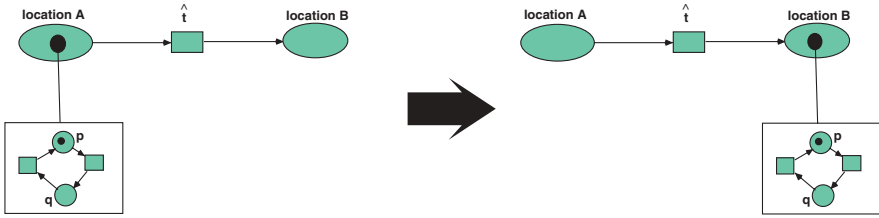


Fig. 4. Transport

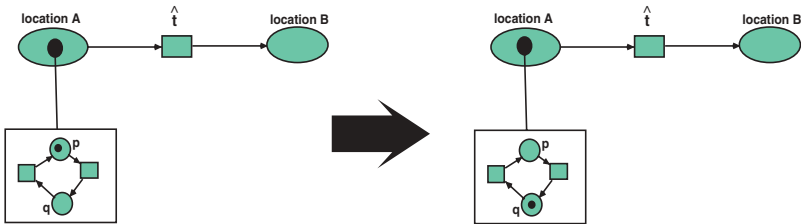


Fig. 5. Autonomous transition

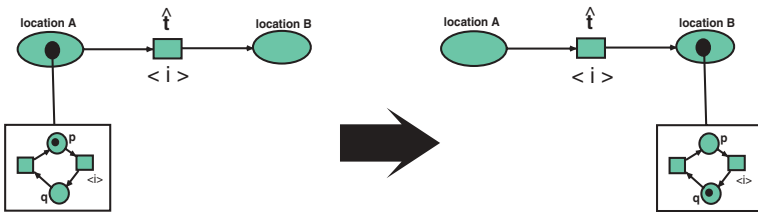


Fig. 6. Interaction

object systems, namely *value semantics* and *reference semantics*. The difference between these semantics becomes obvious when objects (in particular agents) perform concurrent actions at different locations. A single (human) agent can execute independent actions using his two hands. To improve conceivability we prefer to speak of a group of agents, an *agency*. In Fig. 7 such an agency moves from location A to locations B and C. This means that one or more members of the agency are doing so. The abstract net form is given in Fig. 8. The concurrent behaviour of the agency is represented by transitions, labelled $\langle i \rangle$ and $\langle j \rangle$. The question now is, what will be the marking after firing the leftmost transition in the system net? The proposal in Fig. 9 shows references to the object net from both of the output places of the transition. This can be interpreted in such a way, that the members of the agency refer to the same action plan as before, but from different locations. In the graphical representation dashed arrows are used to distinguish references from the lines used before for linking the object nets. As the action plan matches the system net, the concurrent actions labelled $\langle i \rangle$ and $\langle j \rangle$ can be concurrently executed. Later in this paper we discuss how to define semantics for the corresponding join of these action sequences. Distributed systems are characterised by the impossibility of direct access to common data. To

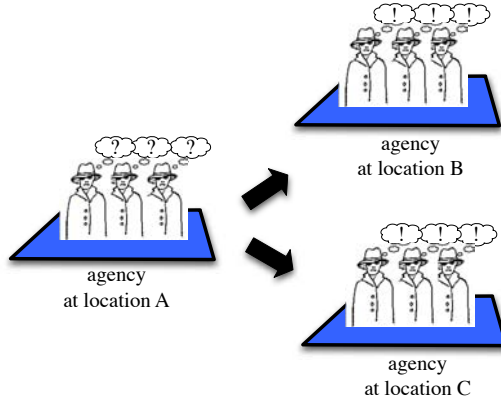


Fig. 7. Creating distributed agencies

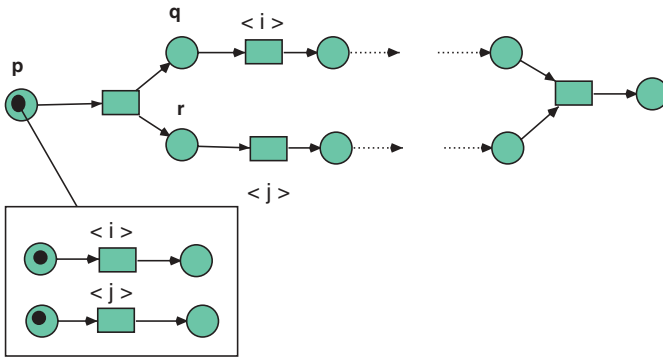


Fig. 8. Creating distributed object nets

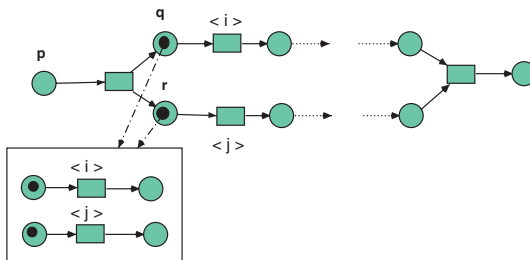


Fig. 9. Reference semantics

meet this paradigm, *value semantics* have been introduced. Instead of references identical copies of the object net are assigned to the output places of the system net transition. This is similar to *call by value* in procedure parameter passing. In the running example, with value semantics, from the marking in Fig. 8 we get

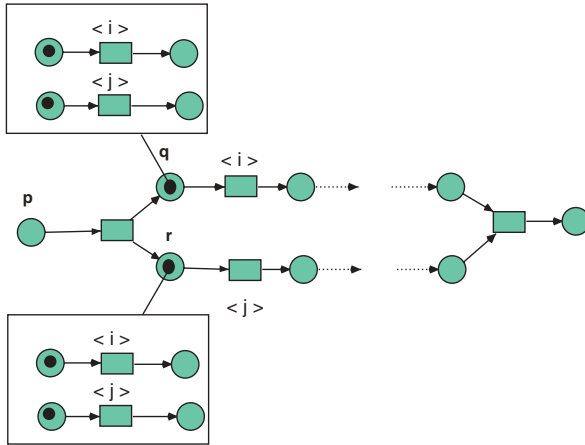


Fig. 10. Value semantics

a successor marking as shown in Fig. 10. Note that normal lines are used again (instead of dashed arrows).

The nets-within-nets concept was first introduced in the nineteen-eighties as *task/flow-nets*, [2–4]. Further results have been published in [5], [6] and [7]. The relation between reference and value semantics is discussed in [8–10]. Reference semantics are carefully studied in the theses of O. Kummer [11] and F. Wienberg [12], in particular in the context of the Renew tool [13]. In the thesis of B. Farwer [14] value semantics is studied within the framework of linear logic (see also [15–19]). Recent work with applications to distributed agents, mobile systems, security problems and socionics can be found in [20–24].

Many references connect Petri net models with object orientation [25–52]. These approaches introduce features of object oriented languages into Petri nets, like classes and inheritance, and partially also refer to the nets-within-nets paradigm.

2 Elementary Object Systems

We now formally introduce *elementary object systems* which form a restricted class of general object systems. We only allow two types (colours) for places, namely objects from a given set of object nets (which do not contain token-nets again) and ordinary black tokens. For general object systems more types are allowed. By this restriction the model remains simple, yet most important features can be introduced. To alleviate the distinction between system and object nets the components of the system net will bear a hat: $\hat{t}, \hat{p}, \hat{P}, \hat{T}, \dots$ etc.

Definition 1. An elementary object system is a tuple $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$ where SN is the system net, \mathcal{ON}_{m^0} is a finite set $\{(ON_1, m_1^0) \dots, (ON_k, m_k^0)\}$ of marked object nets, ϱ is the interaction relation and \mathbf{R}_0 is the initial marking,

which are defined as follows. (The sets of places and transitions of all involved nets are assumed to be finite and disjoint.)

- a) A system net is a Petri net $SN = (\widehat{P}, \widehat{T}, \widehat{W})$ where
 1. the set $\widehat{P} = P_{ob} \cup P_{bt}$ of places is divided into disjoint sets of object places P_{ob} and black-token places P_{bt} , \widehat{T} is the set of transitions,
 2. the set of arrows is given as a mapping $\widehat{W} : (\widehat{P} \times \widehat{T}) \cup (\widehat{T} \times \widehat{P}) \rightarrow \mathbb{N}$. For $\widehat{W}(x, y) > 0$ the arrow (x, y) is called an object arrow if $\{x, y\} \cap P_{ob} \neq \emptyset$ and a black-token arrow if $\{x, y\} \cap P_{bt} \neq \emptyset$.
- b) An object net is a P/T net $ON_i = (P_i, T_i, W_i)$ (cf. the Appendix)¹.
- c) $\varrho \subseteq \widehat{T} \times T$ is the interaction relation where $T := \bigcup_{i=1}^k T_i$.
- d) \mathbf{R}_0 specifies the initial token distribution, where $\mathbf{R}_0 : \widehat{P} \rightarrow \mathbb{N} \cup Bag(\mathcal{ON})$ with $\mathcal{ON} := \{ON_1, \dots, ON_k\}$. It has to satisfy the condition $\mathbf{R}_0(\hat{p}) \in \mathbb{N} \Leftrightarrow \hat{p} \in P_{bt}$.

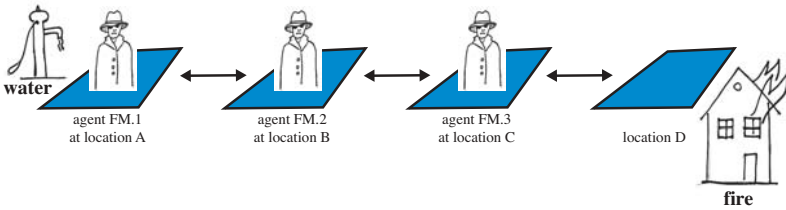


Fig. 11. Parallel fire extinction by agents

In the example of Fig. 12 an object system $OS = (SN, \mathcal{ON}, \varrho, \mathbf{R}_0)$ is shown, where $\mathcal{ON} = \{FM.1, FM.2, FM.3\}$. Black-token arrows of SN can be identified by their labelling from \mathbb{N} . Hence **water** is a black-token place, whereas $\{A, B, C, D\}$ are object-places. In the initial marking places A, B and C contain the object net $FM.1, FM.2$ and $FM.3$, respectively. They have the same structure and could be generated from a type pattern FM . To keep the formal definition simple, we start with all instances of such patterns already generated. The interaction relation is given by corresponding labels in angle brackets:

$$\varrho = \{(\langle \text{refill}, a.1 \rangle, (\text{AtoB}, b.1)), \dots, (\langle \text{refill}, a.2 \rangle, (\text{AtoB}, b.2)), \dots\}$$

(the labels are $\langle \text{refill} \rangle$ and $\langle \text{approachFire} \rangle$ in the given cases).

The formal behaviour of this example object system will be defined in the next section. It is a modification of the well-known *fire extinction example* of C. A. Petri [53]: tree agents (or firemen) $FM.1, FM.2$ and $FM.3$ are in locations A, B and C (Fig. 11). The location D is empty. They can change their position

¹ It is not very important, which class of Petri nets is chosen for object nets. To keep definitions simple, we define them as P/T nets (see Appendix).

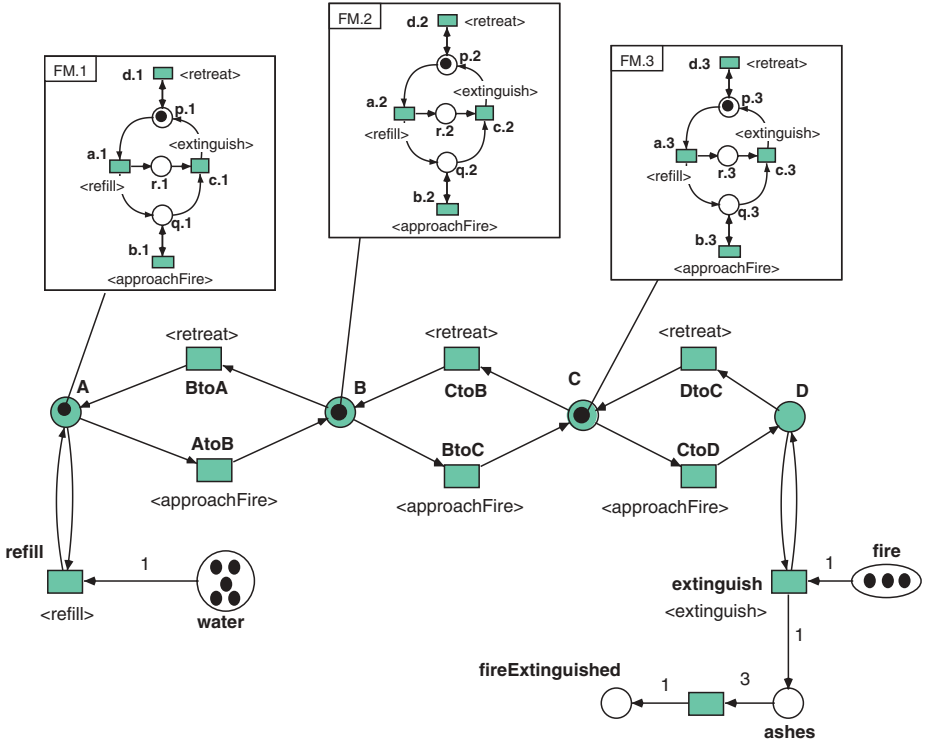


Fig. 12. Elementary object net for parallel fire extinction

to location A, fill their bucket² at a water source, go to location D and help to extinguish the fire. They do this quite independently. In a modification, that will be shown later, they will coordinate their actions to form a chain, like in Petri's setting. To show progress up to termination the amount of water is quantified by 5 black tokens, whereas the fire can be extinguished in 3 steps by removing 3 tokens from the place **fire**. The final marking includes the place **fireExtinguished** and the reader is invited to specify the terminal markings, i.e. where the agents terminate.

3 Reference Semantics of Object Systems

We start by introducing the notion of a marking for object systems under reference semantics. Recall that by Definition 1 an object system contains a set $\mathcal{ON}_{m^0} = \{(ON_1, m_1^0) \dots, (ON_k, m_k^0)\}$ of marked object nets. By omitting the markings we obtain the set of (unmarked) object nets $\mathcal{ON} = \{ON_1, \dots, ON_k\}$. Hence, in general a marking is given by

² The bucket is not modelled here, but in a version given later.

- a) a distribution of object nets or black tokens $\mathbf{R} : \widehat{P} \rightarrow \mathbb{N} \cup \text{Bag}(\mathcal{ON})$ and
 b) the vector $\mathbf{M} = (m_1, \dots, m_k)$ with the current marking of each ON_i ($1 \leq i \leq k$).

\mathbf{R} specifies for each system net place \hat{p} a number of black tokens (if \hat{p} is a black token place) or a multi-set of unmarked object nets (if \hat{p} is an object place). Since the elements of \mathcal{ON} are unmarked, \mathbf{R} can be thought of as a *reference* to the object nets. If we abbreviate (m_1, \dots, m_k) by \mathbf{M} and the set of all such vectors by \mathcal{M} we obtain the following Definition 2. By $pr_i(\mathbf{M})$ we denote the i -th component m_i of \mathbf{M} and by $\mathbf{M}_{i \rightarrow m}$ the tuple, where the i -th component is substituted by m .

Definition 2. *Given an object system $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$ we define $\mathcal{M} := \{\mathbf{M} \mid \mathbf{M} = (m_1, \dots, m_k) \wedge m_i \in \text{Bag}(P_i)\}$. Then a marking of an elementary object system is a pair (\mathbf{R}, \mathbf{M}) where $\mathbf{M} \in \mathcal{M}$ and $\mathbf{R} : \widehat{P} \rightarrow \mathbb{N} \cup \text{Bag}(\mathcal{ON})$ satisfying $\mathbf{R}(\hat{p}) \in \mathbb{N} \Leftrightarrow \hat{p} \in P_{bt}$. Specifying \mathbf{M}_0 by the initial markings of the marked object nets $\mathbf{M}_0 := (m_1^0, \dots, m_k^0)$ we obtain the initial marking $(\mathbf{R}_0, \mathbf{M}_0)$ of OS .*

The occurrence rule for object systems will be introduced in three parts. First we consider the case when an interaction occurs. In this case we assume that a system net transition $\hat{t} \in \widehat{T}$ and an object net transition $t \in T_i$ of some object net ON_i is activated *and* both transitions are related by the interaction relation ϱ . i.e. $(\hat{t}, t) \in \varrho$. This case of the occurrence rule is called an *interaction*.

Definition 3. *(interaction / reference-semantics) Let (\mathbf{R}, \mathbf{M}) be marking of an object system $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$, $\hat{t} \in \widehat{T}$ a transition of SN , $t \in T_i$ a transition of an object net $ON_i = (P_i, T_i, W_i) \in \mathcal{ON}$ such that $(\hat{t}, t) \in \varrho$. Then (\hat{t}, t) is activated in (\mathbf{R}, \mathbf{M}) if:*

- a) $\mathbf{R}(\hat{p}) \geq \widehat{W}(\hat{p}, \hat{t})'ON_i$ for all $\hat{p} \in \bullet\hat{t} \cap P_{ob}$ ³,
 b) $\mathbf{R}(\hat{p}) \geq W(\hat{p}, t)$ for all $\hat{p} \in \bullet\hat{t} \cap P_{bt}$ ⁴ and
 c) t is activated in $m = pr_i(\mathbf{M})$ (see Appendix).

This is denoted by $(\mathbf{R}, \mathbf{M})[\hat{t}, t]$. Let be $m[t]m'$ (w.r.t. ON_i , see Appendix). In this case the successor marking $(\mathbf{R}', \mathbf{M}')$ of OS is defined by

- a) $\mathbf{R}'(\hat{p}) = \mathbf{R}(\hat{p}) - \widehat{W}(\hat{p}, \hat{t})'ON_i + \widehat{W}(\hat{t}, \hat{p})'ON_i$ for all $\hat{p} \in P_{ob}$.
 b) $\mathbf{R}'(\hat{p}) = \mathbf{R}(\hat{p}) - W(\hat{p}, t) + W(\hat{t}, \hat{p})$ for all $\hat{p} \in P_{bt}$.
 c) $\mathbf{M}' = \mathbf{M}_{i \rightarrow m'}$.

This is denoted by $(\mathbf{R}, \mathbf{M})[\hat{t}, t](\mathbf{R}', \mathbf{M}')$.

³ $\widehat{W}(p, t)'ON_i$ denotes the multi-set containing one element ON_i with multiplicity $\widehat{W}(p, t)$. Hence \geq denotes the superset relation of multi-sets.

⁴ \geq denotes the ordering relation of \mathbb{N} .

In our running example of OS from Fig. 12 with $(\mathbf{R}, \mathbf{M}) = (\mathbf{R}_0, \mathbf{M}_0)$, $\hat{t} = \text{refill}$, $ON = FM.1$ and $t = a.1$ we obtain $(\mathbf{R}, \mathbf{M})[\text{refill}, a.1](\mathbf{R}', \mathbf{M}')$ where $\mathbf{R}' = \mathbf{R}$ and $\mathbf{M}' = \mathbf{M}_{1 \rightarrow m_1}$ and $m_1 = \{q.1\}$, i.e. fireman $FM.1$ fills his bucket. In a further step $(\mathbf{R}', \mathbf{M}')[\text{AtoB}, b.1](\mathbf{R}'', \mathbf{M}'')$ a marking $(\mathbf{R}'', \mathbf{M}'')$ is reached where place B contains two token nets, namely $\mathbf{R}''(B) = 1'FM.1 + 1'FM.2$ and $\mathbf{M}'' = (\{q.1\}, \{p.2\}, \{p.3\})$. In this step fireman $FM.1$ approaches the fire by moving to location B .

If a system net transition is activated without being included in the interaction relation, a chosen object net does not change its current marking. As it changes its location in the system net such an occurrence is called a *transport*. The following definition can be seen as the special case of Definition 3 where the involved object net is not changed, i.e. $\mathbf{M}' = \mathbf{M}$ ⁵.

Definition 4. (*transport / reference-semantics*) Let (\mathbf{R}, \mathbf{M}) be a marking of an object system $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$, $\hat{t} \in \hat{T}$ a transition of SN , such that $\hat{t} \notin \text{dom}(\varrho) := \{\hat{t}_1 \mid \exists t : (\hat{t}_1, t) \in \varrho\}$. Then \hat{t} is activated in (\mathbf{R}, \mathbf{M}) if there is an object net ON_i such that

- a) $\mathbf{R}'(\hat{p}) \geq \widehat{W}(\hat{p}, \hat{t})'ON_i$ for all $\hat{p} \in \bullet\hat{t} \cap P_{ob}$,
- b) $\mathbf{R}'(\hat{p}) \geq W(\hat{p}, \hat{t})$ for all $\hat{p} \in \bullet\hat{t} \cap P_{bt}$

Since we use τ for the empty action, this is denoted by $(\mathbf{R}, \mathbf{M})[\hat{t}, \tau]$. In this case the successor marking $(\mathbf{R}', \mathbf{M}')$ is defined by

- a) $\mathbf{R}'(\hat{p}) = \mathbf{R}(\hat{p}) - \widehat{W}(\hat{p}, \hat{t})'ON_i + \widehat{W}(\hat{t}, \hat{p})'ON_i$ for all $\hat{p} \in P_{ob}$.
- b) $\mathbf{R}'(\hat{p}) = \mathbf{R}(\hat{p}) - W(\hat{p}, \hat{t}) + W(\hat{t}, \hat{p})$ for all $\hat{p} \in P_{bt}$.
- c) $\mathbf{M}' = \mathbf{M}$

This is denoted by $(\mathbf{R}, \mathbf{M})[\hat{t}, \tau](\mathbf{R}', \mathbf{M}')$.

In the example of OS from Fig. 12 there is no transport as all system net transitions are labelled for interaction. But we can easily modify the example by deleting all the labels $\langle \text{retreat} \rangle$ in both the system net and all the object nets. Then we obtain the same behaviour as before since there is no different possibility to move for the firemen in the corresponding cases (and their marking did not change anyway). As mentioned in Section 1 object nets may change their state without moving:

Definition 5. (*autonomous action / reference-semantics*) Let (\mathbf{R}, \mathbf{M}) be a marking of an object system $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$ and $t \in T_i$ a transition of an object net $ON_i = (P_i, T_i, W_i) \in \mathbf{R}(\hat{p})$ for some $\hat{p} \in \hat{P}$, such that $t \notin \text{range}(\varrho) := \{t_1 \mid \exists \hat{t} : (\hat{t}, t_1) \in \varrho\}$ and t is activated in ON_i . Then we say that (τ, t) is activated in (\mathbf{R}, \mathbf{M}) (denoted $(\mathbf{R}, \mathbf{M})[\tau, t]$). The successor marking $(\mathbf{R}', \mathbf{M}')$ of OS is defined by

- a) $\mathbf{R}' = \mathbf{R}$.
- b) $\mathbf{M}' = \mathbf{M}_{i \rightarrow m'}$ if $m[t]m'$ for $\text{pr}_i(\mathbf{M}) = m$.

⁵ Definitions 3 and 4 could be easily merged. This is not done to emphasise the differences.

This is denoted by $(\mathbf{R}, \mathbf{M})[\tau, t](\mathbf{R}', \mathbf{M}')$.

Definition 6. For the new alphabet $\Gamma := (\widehat{T} \cup \{\tau\}) \times (T \cup \{\tau\}) \setminus (\tau, \tau)$, where (τ, τ) denotes the neutral element of the free monoid Γ^* , we define:

- a) $(\mathbf{R}, \mathbf{M})[\tau, \tau](\mathbf{R}', \mathbf{M}')$ if $(\mathbf{R}, \mathbf{M}) = (\mathbf{R}', \mathbf{M}')$ and
- b) $(\mathbf{R}, \mathbf{M})[\tilde{w}(\hat{t}, \alpha)](\mathbf{R}', \mathbf{M}')$ if $\exists (\mathbf{R}'', \mathbf{M}'') : (\mathbf{R}, \mathbf{M})[\tilde{w}](\mathbf{R}'', \mathbf{M}'')$ and $(\mathbf{R}'', \mathbf{M}'')[\hat{t}, \alpha](\mathbf{R}', \mathbf{M}')$ for some $\tilde{w} \in \Gamma^*$, $\hat{t} \in \widehat{T} \cup \{\tau\}$ and $\alpha \in T \cup \{\tau\}$.

The examples of transition occurrences given before lead to the following occurrence sequence: $(\mathbf{R}_0, \mathbf{M}_0)[(\text{refill}, \mathbf{a.1}), (\text{AtoB}, \mathbf{b.1})](\mathbf{R}'', \mathbf{M}'')$.

4 Object Interaction, Object Creation and Renew

In this section we extend the definition of elementary object systems to include interaction between objects (with respect to reference semantics). Furthermore we show how these concepts are represented in the Renew tool which includes the creation of object nets.

4.1 Object Interaction

Interaction between object nets is very similar to interaction of system and object nets. But there are good reasons to define them separately. Interacting transitions of different object nets are represented by the interaction relation σ .

Definition 7. The object-object-interaction-relation σ is defined as a set of pairs (t_i, t_j) of transitions t_i and t_j of different object nets ON_i and ON_j . The relation is supposed to be symmetric (i.e. also contains (t_j, t_i)) but irreflexive. Furthermore, to have a simpler formalism it is supposed to be disjoint with the interaction relation ϱ in the following sense: (ϱ, σ) are separated if $(t_1, t_2) \in \sigma \Rightarrow \varrho t_1 = \varrho t_2 = \emptyset$ ⁶.

As for autonomous occurrences object nets may interact without moving. This is restricted, however, to the case where the object nets are locally “near”, which is formalised as *to be in the same place*.

Definition 8. (object-object-interaction / reference-semantics) Let (\mathbf{R}, \mathbf{M}) be a marking of an object system $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$ and $\hat{p} \in \widehat{P}$ a place containing two different object nets $ON_i = (P_i, T_i, W_i)$ and $ON_j = (P_j, T_j, W_j)$, i.e. $ON_i + ON_j \leq \mathbf{R}(\hat{p})$. Then ON_i and ON_j can interact in (\mathbf{R}, \mathbf{M}) if there are transitions $t_i \in T_i$ and $t_j \in T_j$ such that

- a) t_u is activated in $m_u = pr_u(\mathbf{M})$ for both $u \in \{i, j\}$.
- b) $(t_i, t_j) \in \sigma$.

⁶ $\varrho t = \{\hat{t} \mid (\hat{t}, t) \in \varrho\}$

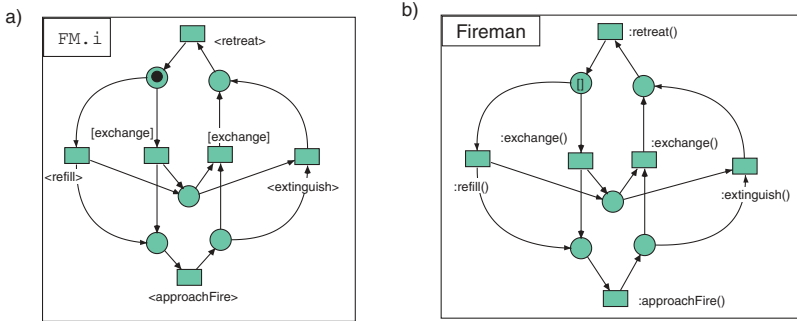


Fig. 13. Fireman with bucket exchange for the net a) of Fig. 12 and b) of Fig. 14

This is denoted $(\mathbf{R}, \mathbf{M})[\tau, t_i | t_j]$. The corresponding successor marking $(\mathbf{R}', \mathbf{M}')$ of OS is defined by

- a) $\mathbf{R}' = \mathbf{R}$.
- b) $\mathbf{M}' = (\mathbf{M}_{i \rightarrow m'})_{j \rightarrow m''}$ if $m_i[t_i]m'$ and $m_j[t_j]m''$.

This is denoted by $(\mathbf{R}, \mathbf{M})[\tau, t_i | t_j](\mathbf{R}', \mathbf{M}')$.

To give an example we substitute the object nets *FM.1*, *FM.2* and *FM.3* from Fig. 12 by three copies of the net from Fig. 13a) for $i = 1, 2, 3$. The modification is the following. Each fireman can proceed only one step by a transition labelled $\langle \text{retreat} \rangle$ or $\langle \text{approachFire} \rangle$. Between these steps there has to be a step with $\langle \text{refill} \rangle$, $\langle \text{extinguish} \rangle$ or $[\text{exchange}]$. Transitions of different object nets labelled by $[\text{exchange}]$ are in the object nets interaction relation σ which is indicated by brackets [and]. Such an interaction can occur only in the “rendez-vous” places *B* and *C*, where they exchange their full and empty buckets. The resulting behaviour is a firemen chain as in Petri’s original example: each fireman moves only between two neighbouring places, whereas the buckets move from the water to the fire and back. The place in the middle of *FM.i* is redundant: if marked the fireman has a full bucket.

4.2 Object Creation and the Renew Tool

The Renew tool allows to design and simulate the example nets in a closely related manner. The system net from Fig. 12 is shown as a Renew model in Fig. 14. There are only little differences. For instance, black tokens are represented by [] and integer weights *n* by *n* such token symbols separated by semi-colons.

The three object nets of the example of Fig. 12 are similar in structure and differ only in the identifiers of places and transitions. The Renew tool supports to define object nets as patterns and to generate instances at simulation runtime. Such a creation is executed by the transition which is placed in the leftmost, upper part of Fig. 14. By the inscription `f1:new Fireman` an instance of the net *Fireman* (see Fig. 13b)) is created with a new identifier *Fireman[n]*, where *n* is

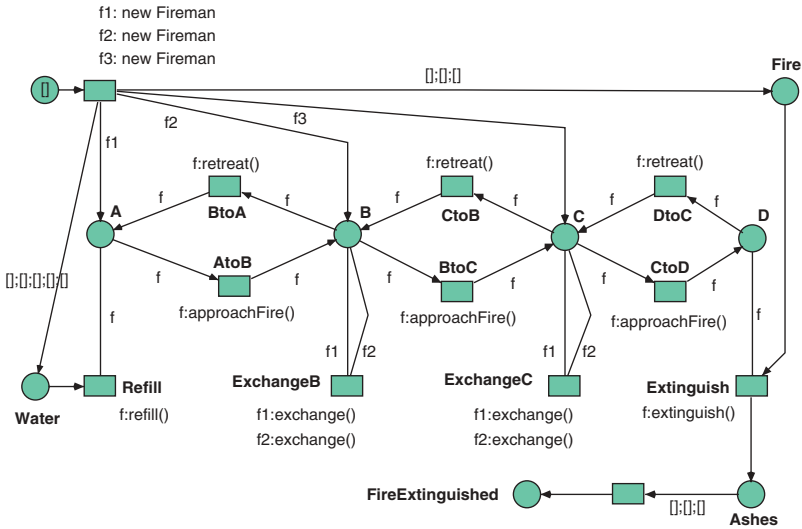


Fig. 14. Fire extinction net modelled with the Renew tool (system net)

an identifier chosen by the tool. Then a reference to this instance is introduced from the place *A* (by the arc labelled $f1$)⁷. In total, the transition creates references to three different object nets in the places *A*, *B* and *C*, 5 black tokens in *Water* and 3 black tokens in *Fire*.

An inscription like f :`approachFire()` is called a *downlink* and is related to a corresponding *uplink* `approachFire()` in the net which is referenced by f (see Fig. 13). The semantics of this pair is the same as for the interaction relation pair $(AtoB, b1) \in \rho$ of Fig. 12. Object-object-interaction is implemented quite different, namely also by down- and uplinks. For such an interaction an extra transition is introduced, like the transition **ExchangeB** in the Renew example net. The downlinks $f1$:`exchange()` and $f2$:`exchange()` contain references to nets in the place *B* (say `Fireman[1]` and `Fireman[2]`) to synchronise two of their transitions labelled by an uplink: `exchange()`. Hence, the behaviour is like the object-object-interaction in the formal definition.

The nets-within-nets paradigm in Renew is not restricted to a 2-level hierarchy: in fact, there is no hierarchy necessary at all. We add a 3-level version of the fire extinction example, where the buckets form an additional level: Fig. 15 and 16. The bucket has two states: `empty` and `full`. The initial state is introduced by the transition with uplink `:new()`, which is executed when the net instance is created (by firing the transition with downlink b :`new Bucket` in the net `Fireman`). It is also interesting to observe how the exchange of a full and an empty bucket is implemented by the transition **ExchangeB** or **ExchangeC**. In the first case the fireman in place *B*, which is referenced by $f1$, executes the transition with uplink `:exchange(be, bf)` whereas a second one, referenced by

⁷ For details see the documentation of the tool at <http://www.renew.de>.

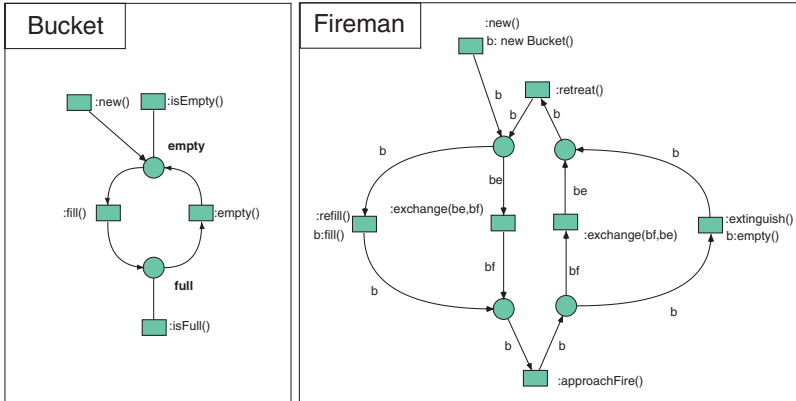


Fig. 15. Three-level fire extinction net modelled with the Renew tool (fireman and bucket net)

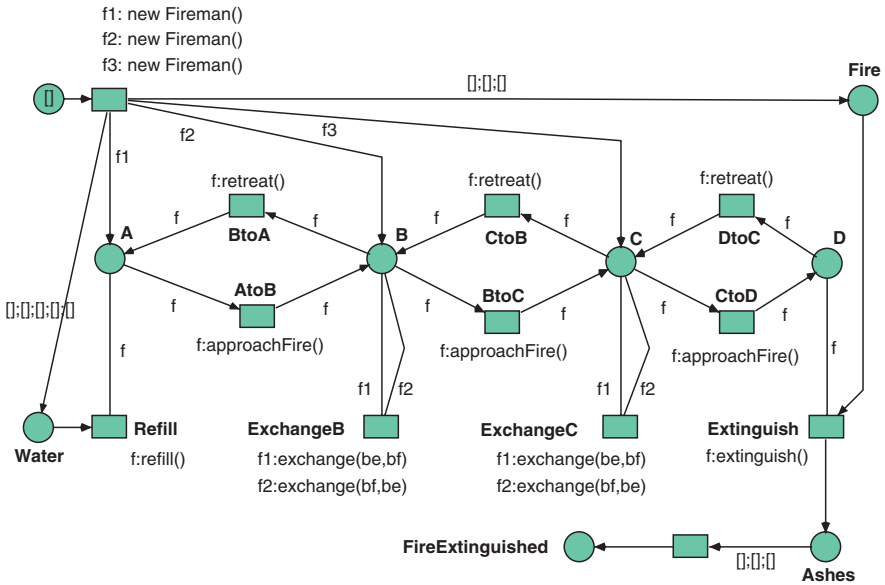


Fig. 16. Three-level fire extinction net modelled with the Renew tool (system net)

f2, does the same for `:exchange(bf, be)`. Hence the empty bucket (referenced by `be`) and the full bucket (referenced by `bf`) are exchanged.

Note that the system nets of the 2-level model (Fig. 14) and the 3-level model (Fig. 16) are very similar, showing the abstraction power of the nets-within-nets modelling paradigm.

5 Value Semantics of Object Nets

In this section value semantics, as introduced in the introduction, will be defined formally. We start with a general definition which will be refined in two following subsections to *distributed tokens semantics* and *history process semantics*. To keep definitions simpler, weights of system nets arrows different to 1 are not considered in this section (i.e. \widehat{W} maps to $\{0, 1\}$ instead of \mathbb{N}).

5.1 Value Semantics of Object Nets: General Definition

In value semantics each object net instance has its own state (marking), which is formalised by the following set \mathcal{ON}_m .

Definition 9. *Given an elementary object system $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$ as in Definition 1 with $\mathcal{ON} = \{ON_1, \dots, ON_k\}$, we now define $\mathcal{ON}_m := \{(ON_1, m_1), \dots, (ON_k, m_k) \mid (m_1, \dots, m_k) \in \mathbf{M}\}$.*

Then a marking of an elementary object system under value semantics is a mapping

$$\mathbf{V} : \widehat{P} \rightarrow \mathbb{N} \cup \text{Bag}(\mathcal{ON}_m)$$

satisfying $\mathbf{V}(\hat{p}) \in \mathbb{N} \Leftrightarrow \hat{p} \in P_{bt}$. The initial marking \mathbf{V}_0 has to meet the condition $\mathbf{V}_0(\hat{p})(ON_i, m_i^0) = \mathbf{R}_0(\hat{p})(ON_i)$, when \hat{p} is an object place and $\mathbf{V}_0(\hat{p}) = \mathbf{R}_0(\hat{p})$ otherwise.

When an interaction occurs with a transition, where several marked object nets are involved at the input side, some kind of unification of their current state (marking) is to be constructed. This corresponds to the collection of partial results (of concurrent computations) to a consistent state, unifying these partial states. The definition of such a function *unify* is left unspecified in the following definition, but made explicit in the subsequent subsections. In a symmetric way, for the output-places a function *distribute* is introduced, which constructs from the state (marking m) a tuple $(m_{\hat{p}_1}, \dots, m_{\hat{p}_q})$ of states (markings) for the object nets to be created in the output places $(\hat{p}_1, \dots, \hat{p}_q)$ of the transition.

Definition 10. (*interaction / value-semantics*) *Let \mathbf{V} be a marking of an elementary object system $OS = (SN, \mathcal{ON}_{m^0}, \varrho, \mathbf{R}_0)$, $\hat{t} \in \widehat{T}$ a transition of SN , $t \in T$ a transition of an object net $ON_i = (P_i, T_i, W_i) \in \mathcal{ON}$ such that $(\hat{t}, t) \in \varrho$. Then (\hat{t}, t) is activated in \mathbf{V} if for each input place $\hat{p} \in \bullet\hat{t} \cap P_{ob}$ there is a submultiset $\mathbf{V}_{\hat{p}} \subseteq \mathbf{V}(\hat{p})$ such that*

- a) $\widetilde{\mathbf{V}}_{\hat{p}} = \widehat{W}(\hat{p}, \hat{t})'ON_i$ for all $\hat{p} \in \bullet\hat{t} \cap P_{ob}$ for all $p \in \bullet t \cap P_{ob}$,
- b) $\mathbf{V}(\hat{p}) \geq W(\hat{p}, \hat{t})$ for all $\hat{p} \in \bullet\hat{t} \cap P_{bt}$ and
- c) $m = \text{unify}(\{m_1 \mid (ON_i, m_1) \in \mathbf{V}_{\hat{p}} \wedge \hat{p} \in \bullet\hat{t} \cap P_{ob}\})$ is defined and t is activated in m , where *unify* is a partial mapping from the set $2^{2^{P_i}}$ of all marking sets of ON_i to the set 2^{P_i} of markings of ON_i .

This is denoted by $\mathbf{V}[\hat{t}, t]$. Let be $m[t]m'$ (w.r.t. ON_i , see Appendix). In this case the successor marking \mathbf{V}' of OS is defined by

- a) $\mathbf{V}'(\hat{p}) = \mathbf{V}(\hat{p}) - \mathbf{V}_{\hat{p}} + \widehat{W}(\hat{t}, \hat{p})'(ON_i, m_{\hat{p}})$ for all $\hat{p} \in P_{ob}$ where $m_{\hat{p}}$ comes from $(m_{\hat{p}_1}, \dots, m_{\hat{p}_q}) \in \text{distribute}(m')$ and $\{\hat{p}_1, \dots, \hat{p}_q\} = \hat{t}^\bullet$. *Distribute* is a mapping from the set 2^{P_i} of markings of ON_i to the set $(2^{P_i})^q$ of all q -tuples of markings of ON_i , where q is the number of output-places of \hat{t} .
- b) $\mathbf{V}'(\hat{p}) = \mathbf{V}(\hat{p}) - W(\hat{p}, \hat{t}) + W(\hat{t}, \hat{p})$ for all $\hat{p} \in P_{bt}$.

This is denoted by $\mathbf{V}[\hat{t}, t]\mathbf{V}'$.

The definitions for transport and autonomous action are similar and omitted here.

5.2 Distributed Tokens Semantics

In distributed tokens semantics the tokens of those object nets, whose copies are distributed to the output-places of a transition are distributed as well, in such a way that they form the original marking when taken all together. Hence instead of Fig. 10 a successor marking like in Fig. 17 is appropriate.

Definition 11. (*interaction / distributed tokens semantics*) *Distributed tokens semantics is obtained by defining the (total) mappings unify and distribute of Definition 10 as follows:*

$$\text{unify}\{m_1, \dots, m_s\} := \sum_{i=1}^s m_i$$

$$\text{and } \text{distribute}(m') := \{(m_1, \dots, m_q) \mid \sum_{i=1}^q m_i = m'\}$$

(Recall that markings are multi-sets and the sum is the multi-set addition.)

In Fig. 18 successor markings of Fig. 17 are shown illustrating the application “unify” of Definition 11. The figure contains two markings, namely before and after the occurrence of the rightmost transition. The selection of tokens from the image of the mapping *distribute* is nondeterministic. There are also selections that are “wrong” in the sense that subsequent occurrences are different or impossible. This feature is much like nondeterministic firing of Petri net transitions in general, where conflict solution is left out of consideration.

5.3 History Process Semantics

A different strategy of token distribution is followed by history process semantics. Here all output transitions are supplied with the same information. Then by the subsequent behaviour the appropriate selection is chosen. In order to check whether concurrent executions are consistent instead of markings, processes (occurrence nets) are used as state information. There is a well-developed theory of processes which is not repeated here (see [54] for instance). We mention that there is a partial order on the set of all processes of a net and a well-defined operation “least upper bound”, which is used in the following definition.

In Fig. 19 the place \hat{p} contains the initial process of the object net (which is omitted in the place). After the occurrence of the interaction (\hat{a}, a) the output-places \hat{q} and \hat{r} are marked with the corresponding enlarged processes. Finally

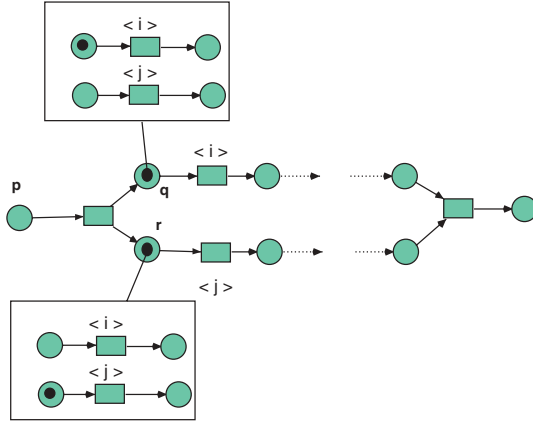


Fig. 17. Successor marking for Fig. 8 with respect to distributed tokens semantics

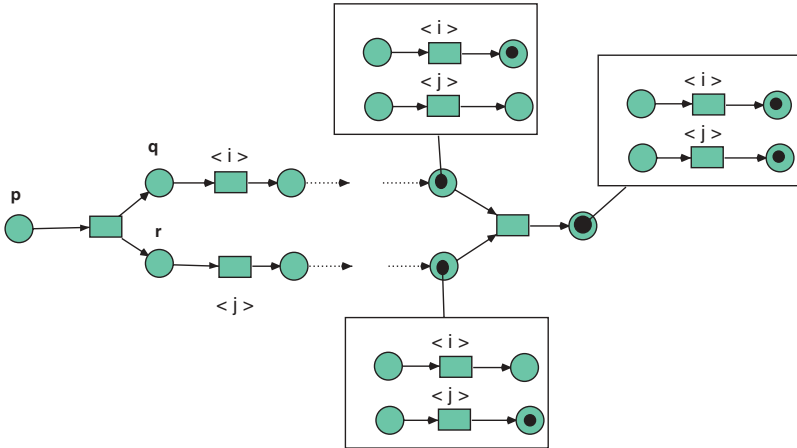


Fig. 18. Successor markings for Fig.17 with respect to distributed tokens semantics

from the processes in \hat{s} and \hat{s}' (see Fig. 20) the least upper bound is constructed and then enlarged by the transition b and added to all output-places of \hat{e} .

Definition 12. (*interaction / history process semantics*) *History process semantics is obtained by defining the mappings unify and distribute of Definition 3 as follows: $\text{unify}\{\text{proc}_1, \dots, \text{proc}_s\} := \bigsqcup_{i=1}^s \text{proc}_i$ and $\text{distribute}(\text{proc}') := \{(\text{proc}', \dots, \text{proc}')\}$, where $\bigsqcup_{i=1}^s \text{proc}_i$ is the least upper bound of all processes proc_i and proc' the least upper bound enlarged by the transition t ⁸.*

⁸ Recall that the transition is not activated, if the least upper bound does not exist.

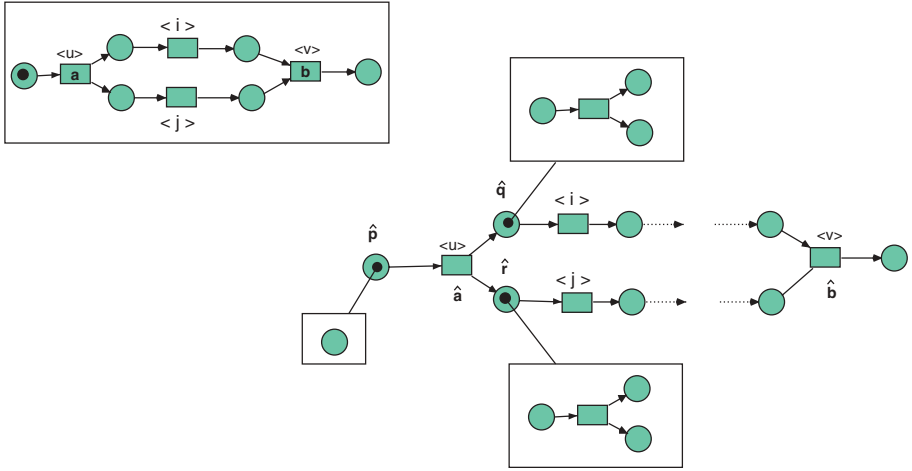


Fig. 19. Initial and first successor markings with respect to history process semantics

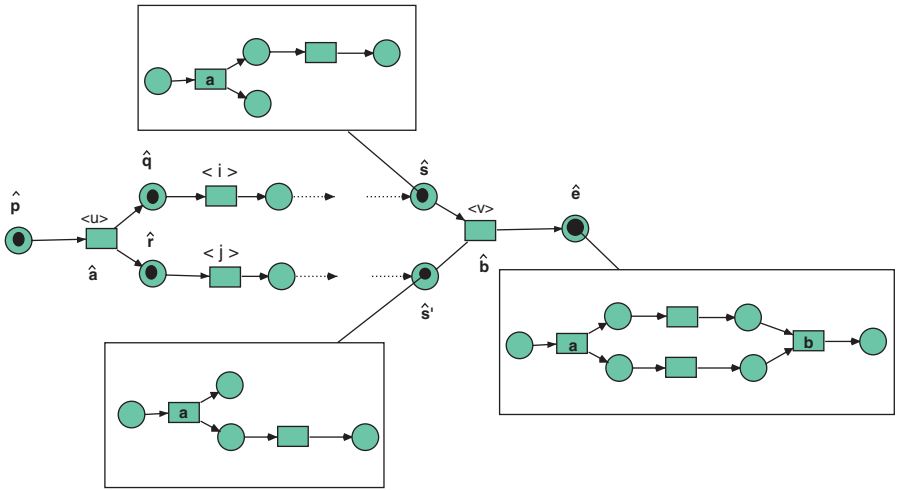


Fig. 20. Successor markings for Fig. 19 with respect to history process semantics

6 Agency under Reference and Value Semantics

In this section we will use a simple example (due to M. Köhler) to explain differences between the introduced semantics. An agent or an agency is designed to first get some money by visiting a bank (3 units of money in our case). Then the agency has to buy flowers uptown (for 1 unit of money) and independently to buy jewels downtown (for 2 units of money). Finally they return from up- and downtown to meet together and deliver their shoppings. We consider 4 different scenarios *system 1* to *system 4* in Fig. 21 to 24. In each of these cases we consider the three introduced semantics *reference semantics* (ref-semantics), *distributed*

tokens value semantics (dt-semantics) and *history process value semantics* (hp-semantics). The behaviour is called *correct* if there is an occurrence sequence that marks exactly the terminal places⁹ in system and object nets, when started with the given initial marking. As discussed above, in particular with dt-semantics there may be correct *and* incorrect behaviours from the same initial marking.

1. Object system 1 (Fig. 21)
 - a) ref-semantics: correct, b) dt-semantics: correct, c) hp-semantics: correct.
2. Object system 2 (Fig. 22)
 - a) ref-semantics: correct, b) dt-semantics: not correct (downtown agency member has no money), c) hp-semantics: not correct (downtown agency member has no money).
3. Object system 3 (Fig. 23)
 - a) ref-semantics: not correct (2 `<visit_bank>`-actions are impossible), b) dt-semantics: not correct (only one agency member has money.), c) hp-semantics: correct.
4. Object system 4 (Fig. 24)
 - a) ref-semantics: not correct (2 `<buy_jewels>`-actions are impossible), b) dt-semantics: not correct (not enough money for both agency members: detected by paying since at least one agency member has not enough money), c) hp-semantics: not correct (not enough money for both agency members: detected by joining since the unify-function is undefined).

Object system 3 (Fig. 23) is of particular interest as this case shows a difference between dt- and hp-semantics. The concept behind hp-semantics is similar to transaction handling in distributed data base systems. Such a transaction is considered *consistent* if it computes consistent results in all sites of the distributed data base system. The `<visit_bank>`-transition is executed uptown *and* downtown, and latter tested on consistency by the last system net transition.

In Fig. 25 two distributed data base systems *DB1* and *DB2* are represented by a simple system net. The object net reads the value of data x (either $x = 0$ or $x = 1$) and terminates with transition `end consistency check`, which in fact behaves like a consistency check under hp-semantics.

7 The Garbage Can Example

This section contains a larger example modelled with the Renew tool. Hence, reference semantics is used. It represents a system of agents that behave partially independent and interact in various ways. This example shows how the nets-within-nets paradigm provides a transparent modelling concept for representing objects of the real world by highly independent, but interacting net instances.

The example comes from a project of *socionics* [22], where knowledge from social sciences and multi-agent systems are combined to profit from each other.

⁹ A place is called *terminal* if it has no output-transitions.

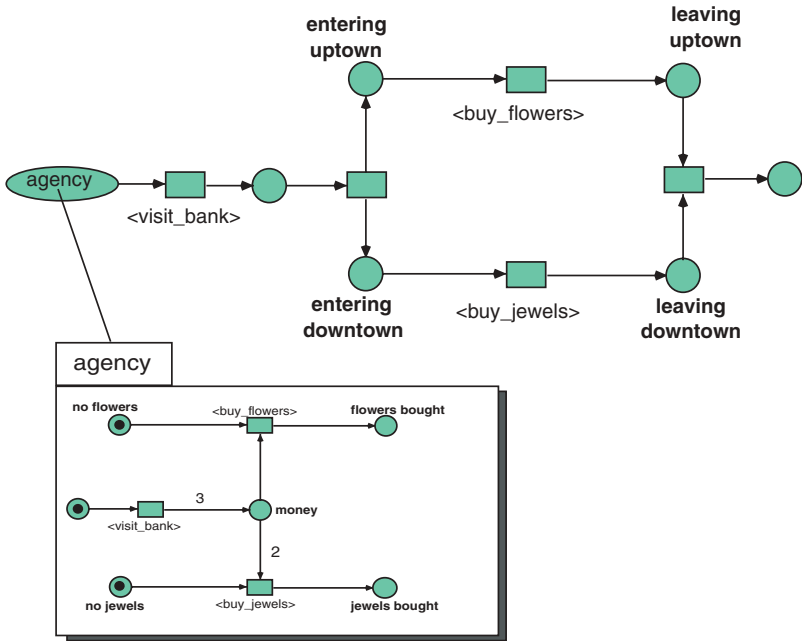


Fig. 21. Object system 1

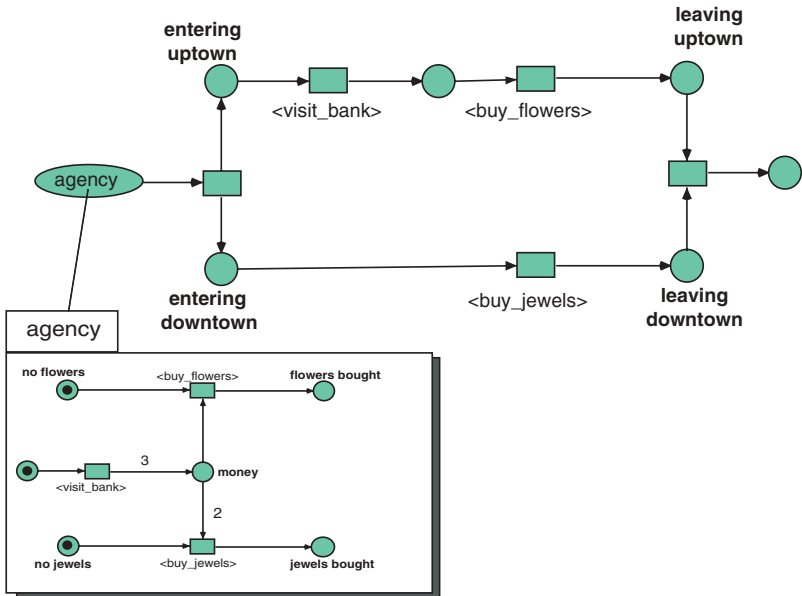


Fig. 22. Object system 2

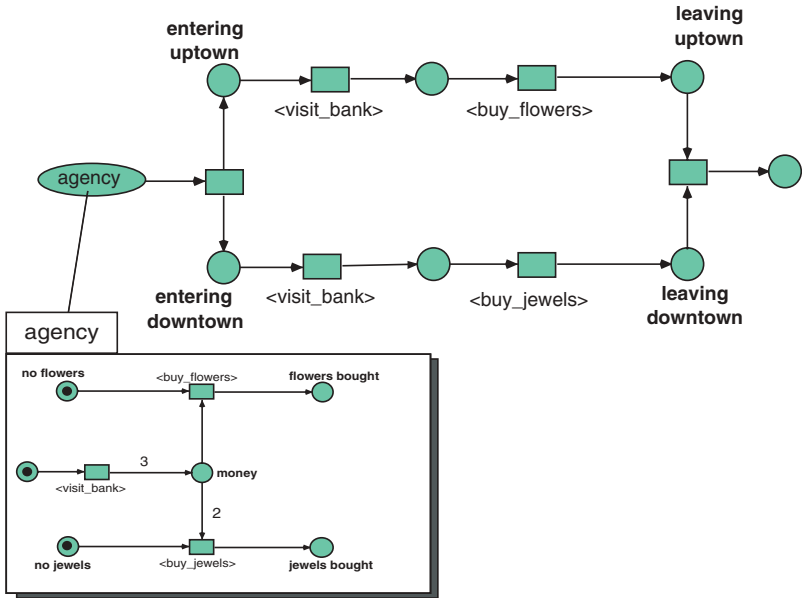


Fig. 23. Object system 3

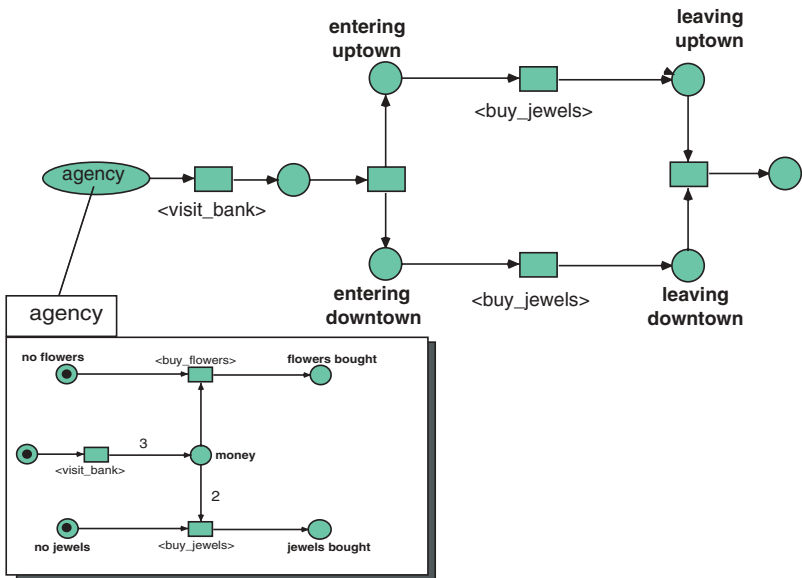


Fig. 24. Object system 4

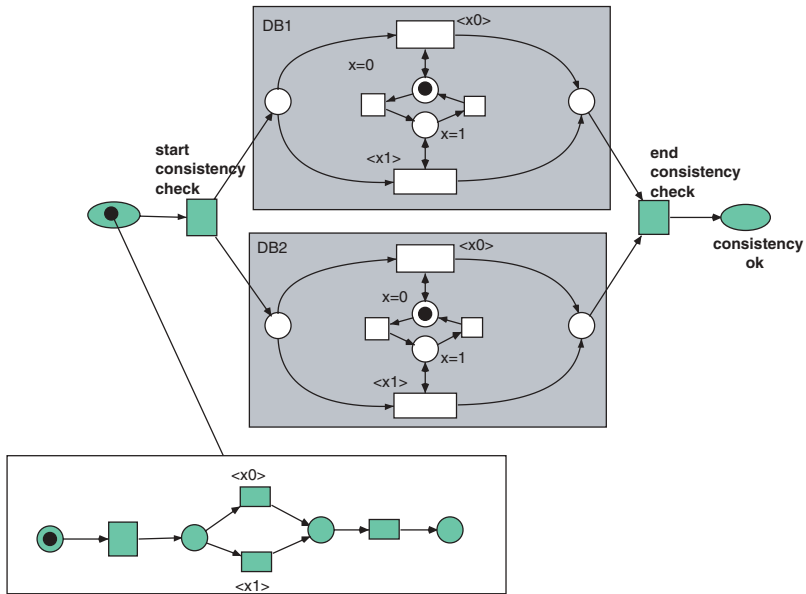


Fig. 25. Consistent distributed reading

It refers to a subfield of organisational theories, where the laws of anarchic behaviour in academic organisations are studied [55]. Following this paper, such organisations - or decision situations - are characterised by three general properties. The first is *problematic preferences*. In the organisation it is difficult to impute a set of preferences to the decision situation that satisfies the standard consistency requirements for a theory of choice. The organisation operates on the basis of a variety of inconsistent and ill-defined preferences. It can be described better as a loose collection of ideas than as a coherent structure; it discovers preferences through action more than it acts on the basis of preferences. The second property is *unclear technology*. Although the organisation manages to survive and even produce, its own processes are not understood by its members. It operates on the basis of simple trial-and-error procedures, the residue of learning from accidents of past experience, and pragmatic inventions of necessity. The third property is *fluid participation*. Participants vary in the amount of time and effort they devote to different domains; involvement varies from one time to another. As a result, the boundaries of the organisation are uncertain and changing; the audiences and decision makers for any particular kind of choice change capriciously.

The authors in [56] distinguish the three notions of *problems*, *solutions*, *participants* and *choice opportunities*. *Problems* are the concern of people inside and outside the organisation. A *solution* is somebody's product. Despite the dictum that you cannot find the answer until you have phrased the question well, you often do not know what the question is in organisational problem solving until you know the answer. Participants come and go. Substantial variation in partic-

ipation stems from other demands on the participants' time (rather than from features of the decision under study). *Choice opportunities* are occasions when an organisation is expected to produce behaviour that can be called a decision. Opportunities arise regularly and every organisation has ways of declaring an occasion for choice. Contracts must be signed; people hired, promoted, or fired; money spent; and responsibilities allocated. The dynamic behaviour is the highly concurrent composition of a stream of choices, a stream of problems, rate of flow of solutions and stream of engaged participants. Where they meet and interact in a unpredictable way is called a *garbage can*.

To get an approximate but more concrete impression of the model, the authors of [56] reconsidered the finale of the James Bond movie, "A View to a Kill". Agent 007 poises on the main cable of the Golden Gate Bridge, a woman in distress clinging to his arm, a blimp approaching for rescue: the blimp is a solution, 007 a choice opportunity, and the woman a problem. In the movie's happy ending, the hero is finally picked up, together with the woman, and a solution by resolution takes place; the problem is solved.

Now imagine numerous blimps, women, and heroes, all arriving out of the blue in random sequence. Heroes take their positions on the main cable. Women cling to heroes, blimps hover above the scene. Heroes are able to hold an unlimited number of women, but the blimp's carrying capacity is limited; heroes with too many women cannot be rescued. Blimps retrieve rescuable, i.e., not-too-heavy, heroes. Women in distress are aware of that and switch heroes opportunistically, choosing the hero closest to retrieval. (In our model, however, women choose heroes nondeterministically.)

Since women, as well as blimps, make their choices simultaneously, but independently of each other, a light hero, on the verge of rescue, may suddenly find himself overburdened. Heavy heroes, in turn, may become rescuable all of a sudden as their women desert them. This mechanism, called "fluid participation", creates the possibility of nonsensical solutions or non-solutions. Women may switch heroes too swiftly and end up with an overburdened hero each time; then, problems are not solved. Or heroes are rescued just as all women have left; then, a "decision by flight" is made. Finally, heroes may be salvaged upon arrival at the scene before any woman in distress has a chance to grab their arm; then "decisions by oversight" are said to be made. Nevertheless, decisions by resolution do occur. Fig. 26 shows the system net, called **bridge**, containing the creation of the agents **woman** (by the annotation **woman:new woman**) and, similarly of the agents **hero** and **blimp**. The modelling of different locations for the **heroes** on the bridge is omitted, but could be easily added. The **heroes** can move to the (common) place **heroes on cable** and are able to cling one or more **women**. By the transition **rescue** they are picked up by a **blimp**, which may continue its flight to the hangar by transition **fly**. As example of a runtime shot in Fig. 26 the instances of agents **woman[1]**, **hero[3]**, **hero[4]** and **blimp[1]** are shown. The instance **blimp[1]** is drawn from the pattern **blimp**, as shown in Fig. 28. Within the place **heroes on cable** there are two instances **hero[3]** and **hero[4]** of the class **hero**. Fig. 27 gives the classes of **hero** and **woman**.

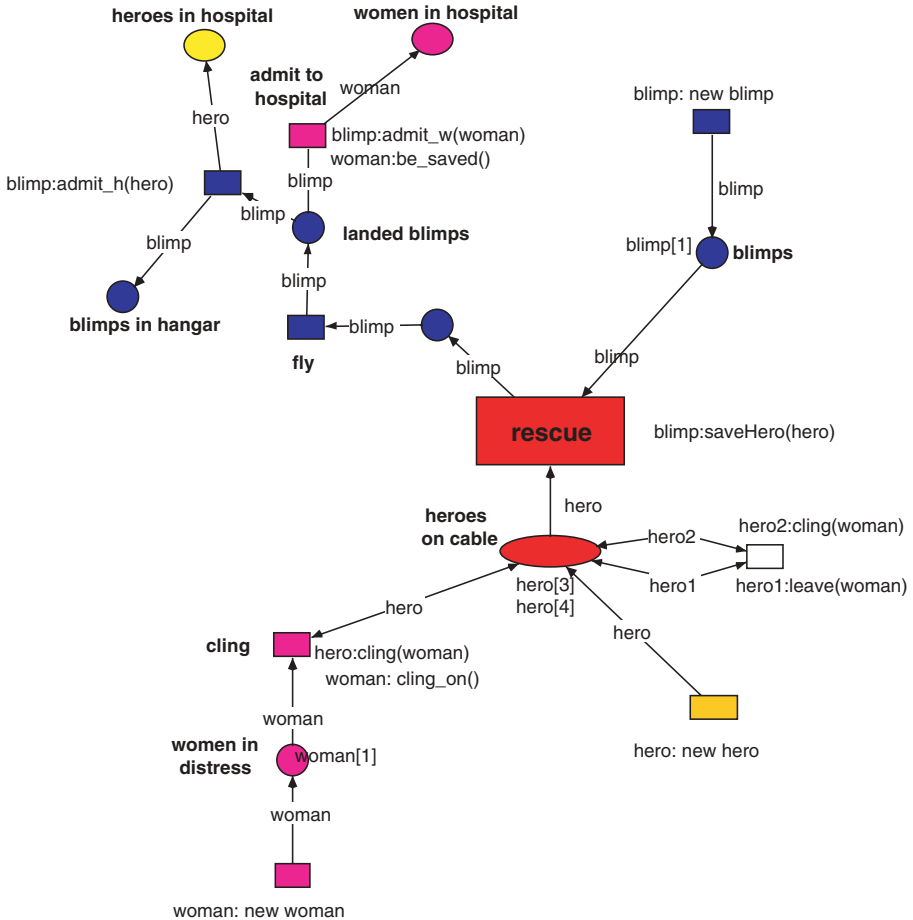
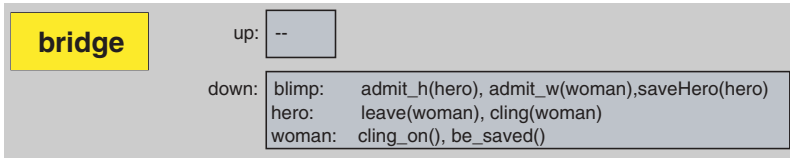


Fig. 26. The garbage can object system: bridge

Each **hero** counts the number of object net instances **woman** picked up by him (see place **counter** in the net **hero**). If there are too many (≥ 3 in our case) transition **rescue** cannot fire for this **blimp** (see leftmost transition of **blimp** in Fig. 28). All down- and uplinks are given in the nets as declarations. This allows for better reading and understanding, but is not supported by the tool. As a modelling style a hierarchy is respected, such that downlinks refer to the next lower level only. By this it is demonstrated how the object-relation *to be contained in* is represented in our modelling style.

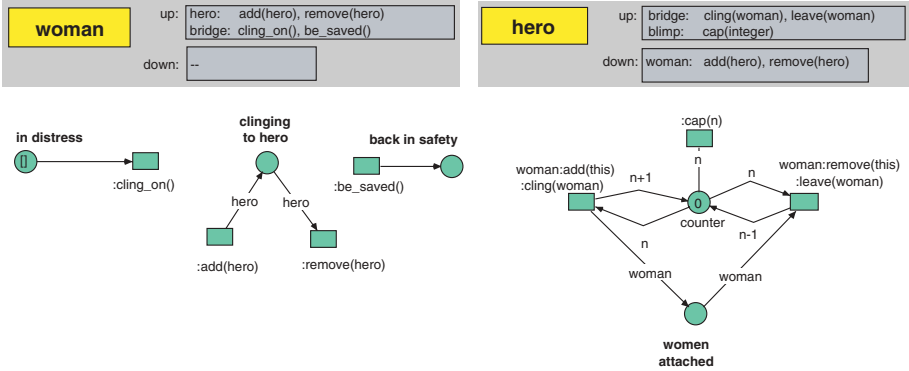


Fig. 27. The garbage can object systems: woman and hero

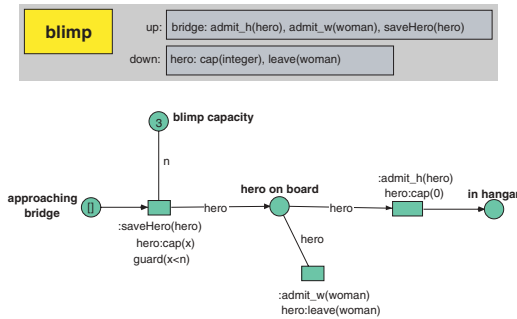


Fig. 28. The garbage can object system: blimp

Note how the change of **woman**[4] from **hero**[3] to **hero**[4] works in this example. **woman**[4] is restricted to communicate only via “her” **hero**[3], who has to forward the procedure call. **hero**[3] has, in turn, also no direct access to **hero**[4]. Instead, he uses the common level **bridge** by the transition **swap**. This mimics reality, where a communication medium is always necessary (e.g. by sight, by mobile phone, by Internet).

This example of garbage cans can be seen as a prototype to other applications of interacting agents, for instance workflow or flexible manufacturing. In the latter case the bridge stands for the machine configuration, the blimps for a conveyors, heroes and women parts to be processed. Furthermore a “production plan” could be added as a further object net, which takes control over the production order, assembly, disassembly etc.

8 Conclusion and Current Research

We have shown that object Petri nets provide a “natural” modelling method, which is easy to understand and is supported by an appropriate tool. The nets-within-nets concept reduces much of the complexity (e.g. readability, simpler arc inscriptions, modular structure) that would result in modelling the same

application by ordinary coloured nets. This is partially a result of the direct representation of object relations like “belongs to” or “is in location”. Furthermore these concepts lead to natural representations of typical properties in distributed systems or mobile computation.

Due to space limitations, it was not possible to present formal results and further modelling examples in application domains. We therefore we give some references to related current research. More definitions, results and examples concerning the concept of history process semantics are given in [6] and [9], whereas distributed tokens semantics are studied in [20, 21, 57–60]. The latter group of references contains results on unbounded marking recursion, concurrency notions and decidability properties of object Petri nets. Modelling mobility and security properties is investigated in [61] and [62]. The bucket-chain example is extended to processor failure (fireman failure) and analysed using the MAUDE tool in [63]. There are also results on pattern based workflow design using reference semantics [64] and a proposal for structuring agent interaction protocols [65]. Model checking for object Petri nets via a translation into Prolog is introduced in [66] while some foundations of dynamic Petri net structures can be found in [67]. Fehling’s concept of hierarchical Petri nets [68] is extended to a class of object Petri nets in [69]. The monograph [22] (in German) reports numerous results on the use of object Petri nets in sociotics (also see [70]). Applications to flexible manufacturing systems can be found in [71] and [72].

Appendix: Basic Concepts

Multi-sets: Let $A \neq \emptyset$ be a set. A *multi-set* s over A is a mapping $s : A \rightarrow \mathbb{N}$, which associates to each element $a \in A$ a non-negative integer coefficient (or multiplicity) $s(a)$. We denote by $Bag(A)$ the set of multi-sets over A . A multi-set will be represented as the symbolic addition of its components: $s = \sum_{a \in A} s(a)'a$. Let s_1 and s_2 be two multi-sets defined over the same set A . The *addition* of multi-sets is defined by $s_1 + s_2 = \sum_{a \in A} (s_1(a) + s_2(a))'a$. On the other hand, $s_1 \geq s_2$ when for each $a \in A$, $s_1(a) \geq s_2(a)$. The *difference* operation extends the corresponding set-operation by $(s_1 - s_2)(a) := \max(s_1(a) - s_2(a), 0)$. For short, \emptyset will be used to denote the *empty multi-set*.

Place/Transition nets: A *Place/Transition net* (*P/T net*) $N = (P, T, W)$ is defined as follows.

1. P and T are finite and disjoint sets of *places* and *transitions*, respectively, and $W : (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$ is the set of weighted arrows.
2. Specifying a marking $m : P \rightarrow \mathbb{N}$ we obtain a *marked P/T net* (N, m_0) . Markings are considered as multi-sets over P .
3. For each $t \in T$ let be $PRE(t)$ and $POST(t)$ the multi-sets over P defined by $PRE(t)(p) := W(p, t)$ and $POST(t)(p) := W(t, p)$, respectively. Then $t \in T$ is *activated* in a marking m if $PRE(t) \leq m$ (denoted $m[t)$) and the transition relation is defined by: $m[t)m' :\Leftrightarrow PRE(t) \leq m \wedge m' = m - PRE(t) + POST(t)$. m' is called *successor marking* of m (w.r.t. t).

References

1. Rumbaugh, J., Jacobson, I., Booch, G.: The unified modeling language reference manual: The definitive reference to the UML from the original designers. Addison-Wesley object technology series. Addison-Wesley, Reading, Mass. (1999)
2. Jessen, E., Valk, R.: Rechensysteme: Grundlagen der Modellbildung. Springer-Verlag, Berlin (1987)
3. Valk, R.: Modelling of task flow in systems of functional units. 124, Universität Hamburg, Fachbereich Informatik (1987)
4. Valk, R.: Nets in computer organisation. In Brauer, W., Reisig, W., Rozenberg, G., eds.: Lecture Notes in Computer Science: Petri Nets: Central Models and Their Properties, Advances in Petri Nets 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, September 1986. Volume 254., Springer-Verlag (1987) 377–396
5. Valk, R.: On processes of object Petri nets. Fachbereichsbericht 185, Fachbereich Informatik, Universität Hamburg (1996)
6. Valk, R.: Petri nets as token objects - an introduction to elementary object nets. In Desel, J., Silva, M., eds.: Application and Theory of Petri Nets 1998, Proceedings 15th International Conference, Lisbon, Portugal. Volume 1420 of Lecture Notes in Computer Science., Springer-Verlag (1998) 1–25
7. Valk, R.: Concurrency in communicating object Petri nets. [73] 164–195
8. Valk, R.: Reference and value semantics for object Petri nets. In: Proceedings of Colloquium on Petri Net Technologies for Modelling Communication Based Systems, October 21–22, 1999, Fraunhofer Gesellschaft, ISST (1999) 169–187
9. Valk, R.: Relating different semantics for object Petri nets, formal proofs and examples. Technical Report FBI-HH-B-226, University of Hamburg, Department for Computer Science Report/00 (2000)
10. Valk, R.: Mobile and distributed object versus central referencing. In Grabowski, Jens, Heymer, Stefan, eds.: Proceedings of 10. GI-ITG-Fachgespräch FBT 2000: Formale Beschreibungstechniken für verteilte Systeme, Lübeck, June 2000, Aachen, Shaker Verlag (2000) 7–27
11. Kummer, O.: Referenznetze. Logos Verlag (2002)
12. Wienberg, F.: Informations- und prozesorientierte Modellierung verteilter Systeme auf der Basis von Feature-Structure-Netzen. Dissertation, Universität Hamburg, Fachbereich Informatik (2001)
13. Kummer, O., Wienberg, F., Duvigneau, M.: Renew – The Reference Net Workshop. <http://renew.de/> (2004)
14. Farwer, B.: Linear Logic Based Calculi for Object Petri Nets. Logos Verlag, ISBN 3-89722-539-5, Berlin (2000)
15. Farwer, B.: Modelling protocols by object-based Petri nets. In Czaja, L., ed.: Concurrency Specification and Programming (CSP'01), Proceedings, University of Warsaw (2001) 87–96 published in Fundamenta Informaticae, 2002.
16. Farwer, B.: Comparing concepts of object Petri net formalisms. Fundamenta Informaticae **47** (2001) 247–258
17. Farwer, B., Lomazova, I.: A systematic approach towards object-based Petri net formalisms. In Bjorner, D., Zamulin, A., eds.: Perspectives of System Informatics, Proceedings of the 4th International Andrei Ershov Memorial Conference, PSI 2001, Akademgorodok, Novosibirsk, LNCS 2244. Springer-Verlag (2001) 255–267
18. Farwer, B.: A multi-region linear logic based calculus for dynamic Petri net structures. Fundamenta Informaticae **43** (2000) 61–79

19. Farwer, B.: A linear logic view of object Petri nets. *Fundamenta Informaticae* **37** (1999) 225–246
20. Köhler, M.: Mobile object net systems: Petri nets as active tokens. Technical Report 320, Universität Hamburg, Fachbereich Informatik, Vogt-Kölln Str. 30, 22527 Hamburg, Germany (2002)
21. Köhler, M., Rölke, H.: Concurrency for mobile object net systems. *Fundamenta Informaticae* **54** (2003)
22. v. Lüde, R., Moldt, D., Valk, R.: *Sozionik: Modellierung soziologischer Theorie*. Lit-Verlag (2003)
23. Cabac, L., Moldt, D., Rölke, H.: A proposal for structuring Petri net-based agent interaction protocols. In van der Aalst, W., Best, E., eds.: *Proc. of 24th International Conference on Application and Theory of Petri Nets 2003 (ICATPN 2003)*, Eindhoven, NL. Volume 2679 of *Lecture Notes in Computer Science.*, Springer-Verlag (2003) 102–120
24. Köhler, M., Moldt, D., Rölke, H.: Modelling mobility and mobile agents using nets within nets. In van der Aalst, W., Best, E., eds.: *Proc. of 24th International Conference on Application and Theory of Petri Nets 2003 (ICATPN 2003)*, Eindhoven, NL, Berlin Heidelberg New York, to be published in *Lecture Notes in Computer Science*, Springer-Verlag (2003)
25. Battiston, E., De Cindio, F., Mauri, G.: Objssa nets: A class of high-level nets having objects as domains. In Rozenberg, G., ed.: *Advances in Petri Nets 1988*. Volume 340 of *Lecture Notes in Computer Science.*, Springer-Verlag (1988) 20–43
26. Battiston, E., Chizzoni, A., Cindio, F.D.: Clown as a testbed for concurrent object-oriented concepts. [73] 131–163
27. Buchs, D., Guelfi, N.: Co-opn: A concurrent object oriented Petri net approach. In: *Application and Theory of Petri Nets, 12th International Conference, IBM Deutschland (1991)* 432–454
28. Christensen, S., Damgaard Hansen, N.: Coloured Petri nets extended with channels for synchronous communication. Technical Report DAIMI PB-390, Aarhus University (1992)
29. Biberstein, O.: CO-OPN/2: An Object-Oriented Formalism for the Specification of Concurrent Systems. Ph.d. thesis, University of Geneva (1997)
30. Biberstein, O., Buchs, D., Guelfi, N.: Object-oriented nets with algebraic specifications: The co-opn/2 formalism. [73] 73–130
31. Lakos, C.A.: Object Petri nets – definition and relationship to coloured Petri nets. Technical Report 94–3, Computer Science Department, University of Tasmania (1993)
32. Lakos, C.A.: From coloured Petri nets to object Petri nets. In: *Proceedings of the 16th International Conference on the Application and Theory of Petri Nets, Turin, Italien*. Volume 935 of *Lecture Notes in Computer Science.*, Springer-Verlag (1995) 278–297 <http://www.cs.adelaide.edu.au/users/charles/>.
33. Lakos, C.A.: The role of substitution places in hierarchical coloured Petri nets. Technical Report 93–7, Computer Science Department, University of Tasmania (1993)
34. Lakos, C.A.: Pragmatic inheritance issues for object Petri nets. In: *Proceedings of TOOLS Pacific 1995, Melbourne, Australien*, Prentice-Hall (1995) 309–321 <http://www.cs.adelaide.edu.au/users/charles/>.
35. Sibertin-Blanc, C.: Cooperative nets. In Valette, R., ed.: *Application and Theory of Petri Nets 1994, Proceedings 15th International Conference, Zaragoza, Spain*. Volume 815 of *Lecture Notes in Computer Science.*, Springer-Verlag (1994) 471–490

36. Sibertin-Blanc, C.: Cooperative objects: Principles, use and implementation. [73] 216–246
37. Sibertin-Blanc, C.: Syroco: A c++ implementation of cooperative objects. Workshop on Petri Nets and Object-Oriented Models of Concurrency (1995) Überarbeitete Version.
38. Project, T.M.: Project homepage. <http://www.tik.ee.ethz.ch/~moses/> (2002)
39. Česka, M., Janoušek, V., Vojnar, T.: Pntalk – a computerized tool for object oriented Petri nets modelling. In Pichler, F., Moreno-Diaz, R., eds.: 6th International Workshop on Computer Aided Systems Theory (EUROCAST'97), Las Palmas de Gran Canaria. Volume 1333 of Lecture Notes in Computer Science., Springer-Verlag (1997) 591–610
40. Goldberg, A., Robinson, D.: Smalltalk-80: The Language. Addison-Wesley (1989)
41. PNTalk: Project homepage. <http://www.fee.vutbr.cz/UIVT/homes/janousek/pntalk/> (2002)
42. Janoušek, V.: Synchronous interactions of objects in object oriented Petri nets. In: Proceedings of MOSIS'99. (1999) 73–80 <http://www.fee.vutbr.cz/~janousek/>.
43. Philippi, S.: OOPr/T-modelle – ein Pr/T-netz basierter Ansatz zur objektorientierten Modellierung. In Desel, J., Oberweis, A., eds.: 6. Workshop Algorithmen und Werkzeuge für Petrinetze, J.W. Goethe-Universität, Institut für Wirtschaftsinformatik, Frankfurt am Main (1999) 36–41
44. Philippi, S.: Seamless object-oriented software development on a formal base. In: Workshop on Software Engineering and Petri-Nets, 21st International Conference on Application and Theory of Petri-Nets, Aarhus. (2000) <http://www.uni-koblenz.de/~philippi/>.
45. Giese, H., Graf, J., Wirtz, G.: Closing the gap between object-oriented modeling of structure and behavior. In France, R., Rumpe, B., eds.: The Second International Conference on The Unified Modeling Language (UML'99). Volume 1723 of Lecture Notes in Computer Science., Springer-Verlag (1999) 534–549
46. OCoN: Project homepage. <http://wwwmath.uni-muenster.de/cs/u/versys/research/ocon/> (2002)
47. Basten, T., van der Aalst, W.M.: Inheritance of behavior. *Journal of Logic and Algebraic Programming* **47** (2001) 47–145
48. Schöpf, S., Sonnenschein, M., Wieting, R.: Efficient simulation of Thor nets. In De Michealis, G., Diaz, M., eds.: Proceeding of the 16th International Conference on Application and Theory of Petri Nets, Turin. Volume 935 of Lecture Notes in Computer Science., Springer-Verlag (1995) 412–431
49. Köster, F., Schöpf, S., Sonnenschein, M., Wieting, R.: Modelling of a library with thorns. [73] 375–390
50. Han, Y.: Software Infrastructure for Configurable Workflow Systems; A Model-Driven Approach Based on Higher-Order Object Nets and Corba. Wissenschaft und Technik Verlag, Berlin (1997) Dissertation an der TU Berlin.
51. Lilius, J.: Ob(pn)²: An object based Petri net programming notation. [73] 247–275
52. Agarwal, R., Bruno, G., Pescarmona, M.: Object-oriented extensions for Petri nets. *Petri Net Newsletter* **60** (2001) 26–41
53. Petri, C.: Introduction to general net theory. In Brauer, W., ed.: *Net Theory and Applications: Proceedings of the Advanced Course on General Net Theory of Processes and Systems*, Hamburg, 1979. Volume 84 of Lecture Notes in Computer Science., Springer-Verlag (1979) 1–19
54. Best, E., Fernández, C.: *Nonsequential Processes. A Petri Net View*. Volume 13. Springer Verlag EATCS Monographs on Theoretical Computer Science (1988)
55. Cohen, M., March, J., Olsen, J.: A garbage can model of organizational choice. *Administrative Science Quarterly* **17** (1972) 1–25

56. Masuch, M., LaPotin, P.: Beyond Garbage Cans: An AI Model of Organizational Choice. *Administrative Science Quarterly* **36** (1989) 38–67
57. Köhler, M., Farwer, B.: Mobile object-net systems and their processes. In: *Proceedings of the International Workshop on Concurrency, Specification, and Programming, CS&P 2003*. (2003) 134–149
58. Köhler, M.: Mobile object net systems. In: *10. Workshop Algorithmen und Werkzeuge für Petrinetze, Universität Eichstätt* (2003) 51–60
59. Köhler, M.: Object Petri nets: Definitions, properties and related models. Technical Report 329, Universität Hamburg, Fachbereich Informatik, Vogt-Kölln Str. 30, 22527 Hamburg, Germany (2003)
60. Köhler, M.: Decidability problems for object Petri nets. In *Gesellschaft für Informatik, ed.: Informatiktage 2003. Fachwissenschaftlicher Informatik-Kongreß, Konradin Verlag* (2003)
61. Köhler, M., Rölke, H.: Modelling sandboxes for mobile agents using nets within nets. In *Busi, N., Martinelli, F., eds.: Workshop on Issues in Security and Petri Nets (WISP'03) at the International Conference on Application and Theory of Petri Nets 2003, University of Eindhoven* (2003)
62. Köhler, M., Moldt, D., Rölke, H.: Modelling mobility and mobile agents using nets within nets. In *v. d. Aalst, W., Best, E., eds.: Proceedings of the International Conference on Application and Theory of Petri Nets 2003. Volume 2679 of Lecture Notes in Computer Science., Springer-Verlag* (2003) 121–140
63. Köhler, M., Rölke, H.: Formal analysis of multi-agent systems: The bucket-chain example. Technical Report to appear, University of Hamburg, Department for Computer Science Report/04 (2004)
64. Moldt, D., Rölke, H.: Pattern based workflow design using reference nets. In *van der Aalst, W., ter Hofstede, A., Weske, M., eds.: Proc. of International Conference on BUSINESS PROCESS MANAGEMENT, Eindhoven, NL, Berlin Heidelberg New York, to be published in Lecture Notes in Computer Science, Springer-Verlag* (2003)
65. Cabac, L., Moldt, D., Rölke, H.: A proposal for structuring petri net-based agent interaction protocols. In: *Lecture Notes in Computer Science: 24th International Conference on Application and Theory of Petri Nets, Eindhoven, Netherlands, June 2003*. (2003)
66. Farwer, B., Leuschel, M.: Model checking object Petri nets in Prolog. Technical Report DSSE-TR-2003-4, Declarative Systems and Software Engineering Group, School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK (2003)
67. Farwer, B., Misra, K.: Dynamic modification of system structures using LLPNs. In: *Perspectives Of System Informatics, Proceedings of the 5th International Andrei Ershov Memorial Conference, PSI 2003, Akademgorodok, Novosibirsk, LNCS 2890. Springer-Verlag* (2003) 274–293
68. Fehling, R.: A concept of hierarchical Petri nets with building blocks. In *Rozenberg, G., ed.: Advances in Petri Nets 1993. LNCS 674, Springer-Verlag* (1993) 148–168
69. Farwer, B., Misra, K.: Modelling with hierarchical object Petri nets. *Fundamenta Informaticae* **55** (2003) 129–147
70. Köhler, M., Moldt, D., Rölke, H., Valk, R.: Structuring of complex socioic systems using reference nets. Technical Report FBI-HH-B-248, University of Hamburg, Department for Computer Science Report/03 (2003)

71. Ezpeleta, J., Moldt, D.: A proposal for flexible testing of deadlock control strategies in resource allocation systems. In Pahlavani, Z., ed.: Proceedings of International Conference on Computational Intelligence for Modelling Control and Automation, in Vienna, Austria, 12-14 February. (2003)
72. Ezpeleta, J., Valk, R.: Modelling assembly systems using object Petri nets and deadlock avoidance. Technical Report to appear, University of Hamburg, Department for Computer Science Report/04 (2004)
73. Agha, G., De Cindio, F., Rozenberg, G., eds.: Advances in Petri Nets: Concurrent Object-Oriented Programming and Petri Nets. Volume 2001 of Lecture Notes in Computer Science. Springer-Verlag (2001)