Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 175 (2007) 119–134

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

# Dynamic Transition Refinement

Michael Köhler and Heiko Rölke

*University of Hamburg, Department for Informatics*

**Abstract**

Refinement of Petri nets is well suited for the hierarchical design of system models. It is used to represent a model at different levels of abstraction.
Usually, refinement is a static concept. For many scenarios, however, it is desirable to have a more flexible form of refinement. For example in the context of service updates, e.g. version control in distributed systems, a mechanism for dynamic transition refinement is needed.
The requirement of dynamic refinement at runtime is quite strong. Since we would like to redefine the system structure by itself, transition refinement cannot be implemented by a model transformation. Instead, an approach is needed which allows for dynamic net structures that can evolve as an effect of transitions firing. In previous work we introduced nets-within-nets as a formalism for the dynamic refinement of tokens. Here we consider an extension of nets-within-nets that uses special net tokens describing the refinement structure of transitions. Using this formalism it is possible to update refinements, introduce alternative refinements, etc. We present some formal properties of the extended formalism and introduce an example implementation for the tool Renew in the context of workflow modeling.

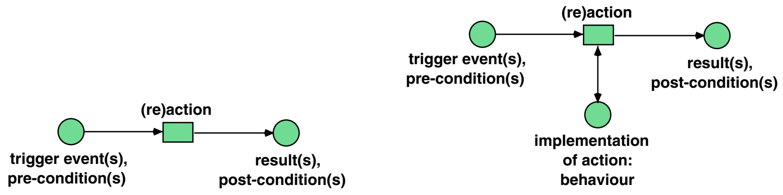
*Keywords:* duality, refinement, nets-within-nets, Petri nets

## 1 Motivation: Refinement of Transitions

The top-down design of a software product has several advantages. Using a graphical formalism for software design, perhaps the most important advantage is to have an abstract, but nevertheless functional, view of the system. A software designer wants to have a (Petri net) view like that in Figure 1(a). To depict a refined – or refinable – transition just like an unrefined one emphasizes the functionality of the abstract net disregarding the refinement. Hence, the net's structure describes the abstract behaviour. At the same time there is the risk to lose sight of the refinement. This may be tolerated if the refinement is free from side effects, but some kind of labelling should be considered.

In later stages of the software design process the (re)action transition in Figure 1(a) should be refined, e.g. by substituting it with one or several sub-nets (perhaps following some refinement rules like those of Fehling [4]). A Petri net formalism supporting transition refinement has the advantage over other formalisms that the original abstract net of the early design stages does not have to be redefined, but is continuously used for the later models and for the implementation models. An

doi:10.1016/j.entcs.2007.03.007


(re)action
trigger event(s), pre-condition(s)
result(s), post-condition(s)
(re)action
trigger event(s), pre-condition(s)
result(s), post-condition(s)
implementation of action: behaviour

(a) A refined transition should look like a normal transition

(b) Idea of dynamic refinement

Fig. 1. First approach to refinement

example of such a formalism is the one of (hierarchical) Coloured Petri Nets [8] used in the Design/CPN tool [2].

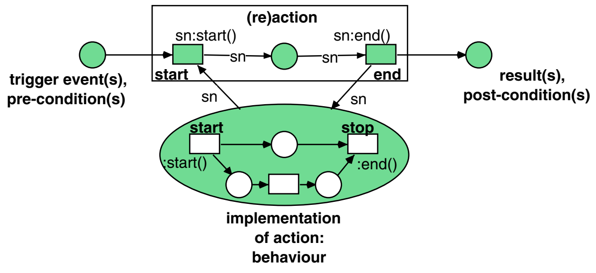

(re)action
sn:start()
sn:end()
sn
sn
start
end
trigger event(s), pre-condition(s)
result(s), post-condition(s)
sn
sn
start
stop
:start()
:end()
implementation of action: behaviour

Fig. 2. Dynamic refinement using reference nets

Unfortunately the proposed transition refinement procedures in [8] only support static refinements that cannot be changed at runtime. In some situations it is desirable to have a *dynamic* refinement at hand [1], for example by specifying the refined net as a marking on a place. This is illustrated in Figure 1(b). The side condition in the figure is meant to specify the refinement mode of the transition, it can be considered as a kind of parameter. Two advantages are met by this approach: First, the refinement is introduced without extra constructs but with well-known net elements, and second, the dynamics may also be expressed in terms of a well-known Petri net action, the firing of transitions altering the marking of the side condition (not illustrated in the figure).

In fact, such a Petri net formalism exists, namely reference nets [13,14], a variant of the nets-within-nets idea of Valk [19]. The desired functionality of Figure 1(b) is illustrated in more detail in Figure 2 using proper syntax of reference nets. [2] The white transition-like box (re)action in the figure only highlights a component and is not to be confused with a transition.

The example of Figure 2 incorporates dynamic refinement, several alternative refining nets are possible, and the change of the refinement can take place at runtime, but it has the disadvantage of changing the graphical structure of the original net

[1] This holds true for all kinds of dynamic systems that should be adapted like workflow systems or that have the ability to adapt theirselves, like multi-agent systems

[2] Note that nets-within-nets actually describe refinements of tokens or states. Because the token refinement is done in terms of an active concept – a net – it has become quite common to use the token refinement as a sort of net structure refinement, and therefore behavior refinement.

– compare Figure 2 to Figure 1(a) – thus making the abstract net view difficult to read. The general idea that has led us to the new net formalism described in this paper was to incorporate the refined net(s) describing the behaviour of a transition (back) into the transition itself, leading to the diagrammatic illustration of Figure 3.

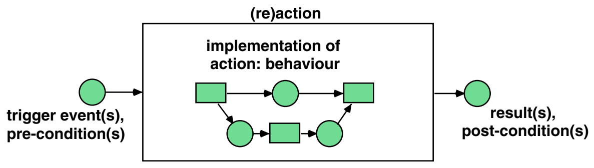

(re)action
implementation of action: behaviour
trigger event(s), pre-condition(s)
result(s), post-condition(s)

Fig. 3. Dynamic transition refinement

The "only" novelties our new formalism requires, are *marked transitions* and the *firing of places*(sic!). [3]

The remaining sections are structured as follows: In Section 2 we introduce our approach of marked transitions for Petri nets. We define super-dual nets and their firing rule. In Section 3 we describe how the concept of super-dual nets can be lifted to object nets and give an abbreviated definition of the new formalism of super-dual object nets. In Section 4 we describe how super-dual object nets can be simulated by object nets. Section 5 explains a first attempt to integrate dynamic transition refinement in Renew. The paper ends with a conclusion. The appendix recalls definitions of multi-sets.

## 2 Introduction to Super-Dual Petri Nets

This section starts with a short remainder of Petri nets basics. This is to avoid notational confusions. After that super-dual nets will be introduced.

### 2.1 Basic Definitions

The definition of Petri nets relies on the notion of multi-sets. A multi-set on the set $D$ is a mapping $\mathbf{A} : D \to \mathbb{N}$. The set of all mapings from $D$ to $\mathbb{N}$ is denoted by $\mathbb{N}^D$. Multi-sets are generalisations of sets in the sense that every subset of $D$ corresponds to a multi-set $\mathbf{A}$ with $\mathbf{A}(x) \leq 1$ for all $x \in D$. The empty multi-set $\mathbf{0}$ is defined as $\mathbf{0}(x) = 0$ for all $x \in D$. The carrier of a multi-set $\mathbf{A}$ is $\mathrm{dom}(\mathbf{A}) := \{x \in D \mid \mathbf{A}(x) > 0\}$. The cardinality of a multi-set is $|\mathbf{A}| := \sum_{x\in D} \mathbf{A}(x)$. A multi-set $\mathbf{A}$ is called *finite* iff $|\mathbf{A}| < \infty$. The multi-set sum $\mathbf{A}+\mathbf{B}$ is defined as $(\mathbf{A}+\mathbf{B})(x) := \mathbf{A}(x)+\mathbf{B}(x)$, the difference $\mathbf{A} - \mathbf{B}$ by $(\mathbf{A} - \mathbf{B})(x) := \max(\mathbf{A}(x) - \mathbf{B}(x), 0)$. Equality $\mathbf{A} = \mathbf{B}$ is defined element-wise: $\forall x \in D : \mathbf{A}(x) = \mathbf{B}(x)$. Multi-sets are partially ordered: $\mathbf{A} \leq \mathbf{B} \iff \forall x \in D : \mathbf{A}(x) \leq \mathbf{B}(x)$. The strict order $\mathbf{A} < \mathbf{B}$ holds iff $\mathbf{A} \leq \mathbf{B}$ and $\mathbf{A} \neq \mathbf{B}$. The notation is overloaded, being used for sets as well as multi-sets. The meaning will be apparent from its use.

In the following we assume all multi-sets to be finite. A finite multi-set $\mathbf{A}$ can be considered as the formal sum $\mathbf{A} = \sum_{x\in D} \mathbf{A}(x) \cdot x = \sum_{i=1}^{n} x_i$. Finite multi-sets

[3] Note, that in Figure 3 (re)action is a proper transition with the implementing net as a marking.

are the freely generated commutative monoid. If the set $D$ is finite, then a multi-set $\mathbf{A} \in \mathbb{N}^D$ can be represented equivalently as a vector $\mathbf{A} \in \mathbb{N}^{|D|}$.

Any mapping $f : D \to D'$ can be generalised to a homomorphism $f^\sharp : \mathbb{N}^D \to \mathbb{N}^{D'}$ on multi-sets: $f^\sharp\left(\sum_{i=1}^n a_i\right) = \sum_{i=1}^n f(a_i)$. This includes the special case $f^\sharp(\mathbf{0}) = \mathbf{0}$. These definitions are in accordance with the set-theoretic notation $f(A) = \{f(a) \mid a \in A\}$. In this paper we simply use $f$ instead of $f^\sharp$.

## 2.2 Petri Nets

A *Petri net* is a tuple $N = (P, T, F)$ where $P$ is a set of places, $T$ is a set of transitions, disjoint from $P$, i.e. $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation. Some commonly used notations for Petri nets are ${}^\bullet y := (\_F\, y)$ for the *preset* and $y^\bullet := (y\, F\_)$ for the *postset* of a net element $y$.

To simplify the definition of duality and conjugation we only consider ordinary Petri nets, i.e. we do not deal with arc weights. The mappings $F^-, F^+$ are defined by $F^-(t)(p) := |F \cap \{(p,t)\}|$ and $F^+(t)(p) := |F \cap \{(t,p)\}|$.

A marking of a net $N$ is a multi-set of places: $\mathbf{m} \in \mathbb{N}^P$. Places are depicted as circles, transitions as rectangles, and the flow relation as arcs between the nodes. The marking is visualised as $\mathbf{m}(p)$ tokens on the place $p$.

A *marked Petri net* is a tuple $(N, \mathbf{m}_0)$ consisting of a Petri net and a start marking. Throughout this paper we speak of *Petri nets* or simply *nets* instead of (ordinary) *marked Petri nets.*

A multi-set of transitions $\mathbf{u} \in \mathbb{N}^T$ of a net $N$ is *enabled* in marking $\mathbf{m}$ iff $\forall p \in P : \mathbf{m}(p) \geq F^-(\mathbf{u})(p)$ holds. The enablement of $\mathbf{u}$ in marking $\mathbf{m}$ is denoted by $\mathbf{m} \xrightarrow{\mathbf{u}}$. A transition multi-set $\mathbf{u}$ enabled in $\mathbf{m}$ can fire in the successor marking $\mathbf{m}'$ where $\mathbf{m}'(p) = \mathbf{m}(p) - F^-(u)(p) + F^+(u)(p)$. Firing is denoted by $\mathbf{m} \xrightarrow{\mathbf{u}} \mathbf{m}'$.

Using multi-set operators $\mathbf{m} \xrightarrow{\mathbf{u}}$ is equivalent to $\mathbf{m} \geq F^-(\mathbf{u})$, and the successor marking is $\mathbf{m}' = \mathbf{m} - F^-(\mathbf{u}) + F^+(\mathbf{u})$.

## 2.3 Super-dual Nets

A super-dual net contains a $G$-flow (short: a glow) $G \subseteq (P \times T \cup T \times P)$ as an additional structure. $G$ connects places and transitions the same way as the flow $F$, but with a different semantics (see below).

**Definition 2.1** A *super-dual net* is a tuple $SD = (P, T, F, G)$ where

- $P$ is a set of places,
- $T$ is a set of transitions with $P \cap T = \emptyset$,
- $F \subseteq (P \times T \cup T \times P)$ is the flow relation, and
- $G \subseteq (P \times T \cup T \times P)$ is the glow relation.

The preset w.r.t. the glow $G$ is ${}^\blacksquare y := (\_G\, y)$ and the postset is $y^\blacksquare := (y\, G\_)$. Analogously to the flow mappings we define the glow mappings $G^-, G^+ : T \to (P \to \mathbb{N})$ by $G^-(t)(p) := |G \cap \{(p,t)\}|$ and $G^+(t)(p) := |G \cap \{(t,p)\}|$.

In super-dual nets also the transitions may be marked. A marking of a super-dual net is a multi-set of places and transitions: $\mathbf{m} \in \mathbb{N}^{(P \cup T)}$. The tokens on transitions are called *pokens*. A poken is visualised as a little filled square. A marked super-dual net is denoted as $(P, T, F, G, \mathbf{m})$.

For super-dual nets the firing rule considers the firing of transitions as well as the firing of places.

(i) A marking $\mathbf{m}$ enables a transition $t$ only if its preset ${}^\bullet t$ is marked and $t$ itself is marked. For a transition multi-set $\mathbf{u} \in \mathbb{N}^T$ we define enabling by:

$$\begin{array}{ll} \mathbf{m}(p) \geq F^-(\mathbf{u})(p) & \text{for all } p \in P \\ \mathbf{m}(t) \geq \mathbf{u}(t) & \text{for all } t \in T \end{array}$$

This means, that the number of pokens $\mathbf{m}(t)$ limits the maximal concurrency of the transition $t$. Thus $\mathbf{m}(t) = 0$ describes a disabled transition.

(ii) Conversely, a marking $\mathbf{m}$ enables a place $p$ only if its preset ${}^{\blacksquare} p$ is marked and $p$ itself is marked. For a place multi-set $\mathbf{u} \in \mathbb{N}^P$ we define enablement by:

$$\begin{array}{ll} \mathbf{m}(p) \geq \mathbf{u}(p) & \text{for all } p \in P \\ \mathbf{m}(t) \geq G^-(\mathbf{u})(t) & \text{for all } t \in T \end{array}$$

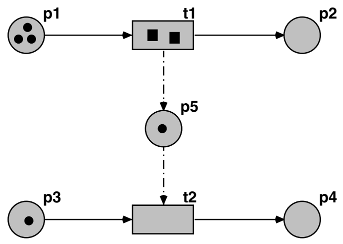


Fig. 4. The super-dual net *SD*

**Example 2.2** Cf. the net in Figure 4. The place $p_5$ is connected by glow arcs (the dashed ones) with transitions $t_1$ and $t_2$. In the depicted marking only the transition $t_1$ is enabled – more exactly: it is enabled twice. Despite the fact, that the preset of transition $t_2$ is marked, it is not enabled, since $t_2$ itself is unmarked. Firing of $p_5$ transfers a poken from $t_1$ to $t_2$, and $t_2$ is then enabled.

Both cases – firing of transitions and of places – can occur in a single step.

**Definition 2.3** A multi-set of places and transitions $\mathbf{u} \in \mathbb{N}^{(P \cup T)}$ of a super-dual net *SD* is *enabled* in the marking $\mathbf{m} \in \mathbb{N}^{(P \cup T)}$, denoted by $\mathbf{m} \xrightarrow{\mathbf{u}}$, iff

$$\begin{array}{ll} \mathbf{m}(p) \geq F^-(\mathbf{u}|_T)(p) + \mathbf{u}(p) & \text{for all } p \in P \text{ and} \\ \mathbf{m}(t) \geq G^-(\mathbf{u}|_P)(t) + \mathbf{u}(t) & \text{for all } t \in T. \end{array}$$

An enabled multi-set $\mathbf{u}$ can fire, denoted by $\mathbf{m} \xrightarrow{\mathbf{u}} \mathbf{m}'$, resulting in the successor marking $\mathbf{m}'$ defined by

$$\mathbf{m}'(p) = \mathbf{m}(p) - F^-(\mathbf{u}|_T)(p) + F^+(\mathbf{u}|_T)(p)$$
$$\mathbf{m}'(t) = \mathbf{m}(t) - G^-(\mathbf{u}|_P)(t) + G^+(\mathbf{u}|_P)(t).$$

Define $\mathbf{pre}(\mathbf{u}) := F^-(\mathbf{u}|_T) + G^-(\mathbf{u}|_P)$ and $\mathbf{post}(\mathbf{u}) := F^+(\mathbf{u}|_T) + G^+(\mathbf{u}|_P)$. Using multi-set notations $\mathbf{m} \xrightarrow{\mathbf{u}}$ is equivalent to $\mathbf{m} \geq \mathbf{pre}(\mathbf{u}) + \mathbf{u}$. The successor marking is $\mathbf{m}' = \mathbf{m} - \mathbf{pre}(\mathbf{u}) + \mathbf{post}(\mathbf{u})$.

*Duality*

Given a super-dual net $SD = (P, T, F, G, \mathbf{m})$ the dual net (interchanging transitions and places) is defined as $SD^d := (T, P, F, G, \mathbf{m})$ and the conjugated net (interchanging flow and glow) is $SD^c := (P, T, G, F, \mathbf{m})$. Note, that also the dual of a marking can be considered for super-dual nets. We have the commutativity: $SD^{cd} = SD^{dc}$. For the super-dual net of Figure 4 these constructions are illustrated in Figure 5.

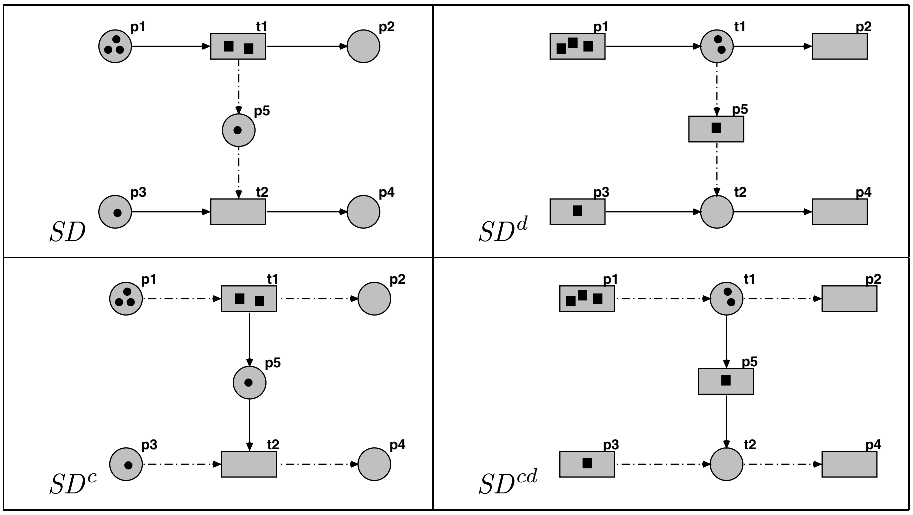


Fig. 5. Duality and Conjugation

The following property justifies the name "super-dual nets".

**Proposition 2.4** *Let SD be a super-dual net. SD corresponds to* $SD^{cd}$:

$$\mathbf{m} \xrightarrow[SD]{\mathbf{u}} \mathbf{m}' \iff \mathbf{m} \xrightarrow[SD^{cd}]{\mathbf{u}} \mathbf{m}'$$

**Proof.** Simultanously interchanging $P$ and $T$ as well as $F$ and $G$ in Definition 2.3 is the identity transformation. □

*Components*

We define the $F$-component $SD|_F$ and the $G$-component $SD|_G$ of a marked super-dual net $SD = (P, T, F, G, \mathbf{m})$ as:

$$SD|_F := (P, T, F, \mathbf{m}|_P) \tag{1}$$

$$SD|_G := (P, T, G, \mathbf{m}|_T) \tag{2}$$

Both constructions are illustrated in Figure 6. Note, that the components $SD|_F$ and the dual of the $G$-component, $SD|_G^d$ (but not $SD|_G$ itself) are Petri nets.

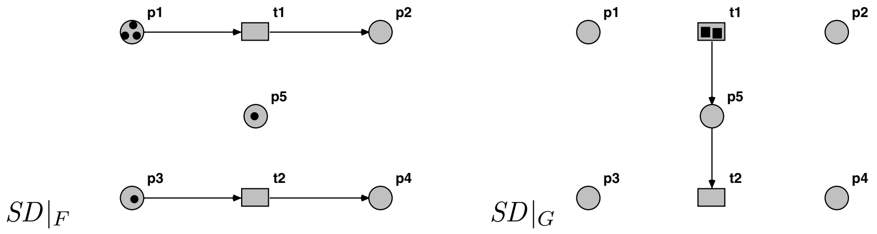


Fig. 6. The $F$- and the $G$-component

The following proposition relates the behaviour of a super-dual net to that of its components.

**Proposition 2.5** *Let* $\mathbf{m} \in \mathbb{N}^{(P \cup T)}$ *be a marking of a super-dual net* $SD = (P, T, F, G)$*. Let* $\mathbf{u} \in \mathbb{N}^{(P \cup T)}$*. Then for the* $F$*-component and the dual of the* $G$*-component we have (For the proof see [12]):*

$$\forall \mathbf{u} \in \mathbb{N}^T : \mathbf{m} \xrightarrow[SD]{\mathbf{u}} \mathbf{m}' \iff \left( \mathbf{m}|_P \xrightarrow[SD|_F]{\mathbf{u}} \mathbf{m}'|_P \wedge \mathbf{m}|_T = \mathbf{m}'|_T \geq \mathbf{u} \right)$$

$$\forall \mathbf{u} \in \mathbb{N}^P : \mathbf{m} \xrightarrow[SD]{\mathbf{u}} \mathbf{m}' \iff \left( \mathbf{m}|_T \xrightarrow[SD|_G^d]{\mathbf{u}} \mathbf{m}'|_T \wedge \mathbf{m}|_P = \mathbf{m}'|_P \geq \mathbf{u} \right)$$

## 3 From Object Nets to Super-Dual Object Nets

We are interested in a dynamic refinement of transitions, i.e. a refinement that can be changed at runtime. This change should be made by the net itself. Our basic approach is to regard sub-nets as special tokens of transitions. As mentioned in the introduction this approach proposes two extensions to the Petri net formalism: (1) Petri nets can be used as tokens and (2) transitions may be marked.

The first extension of Petri nets to object nets – also known as the *nets-within-nets* approach – has been proposed by Valk [18,19], and further developed e.g. in [3], [16], and [10,11]. The Petri nets that are used as tokens are called *net-tokens*. Net-tokens are tokens with internal structure and inner activity. This is different from place refinement, since tokens are transported while a place refinement is static. Net-tokens are some kind of *dynamic* refinement of states.

## 3.1 Object Nets and Object Net Systems

In the following we give a condensed definition of object net systems. For simplicity reasons we abstract from the syntax of inscriptions of net elements and synchronisations as it is used for reference nets [13] in the Renew tool (cf. [14]).

Object net systems have the set of object nets as their colour set. In [10] we generate the net-tokens via instantiation from a finite set of nets. In this definition, we assume for simplicity reasons an arbitrary set of object nets:

$$\mathcal{N} = \{N_0, N_1, \ldots\}$$

One object net models black tokens: $\bullet \in \mathcal{N}$. This net has one initally unmarked place and no transitions.

In coloured nets each transition $t$ fires according to a mode $b$ generated from transition guards, arc expressions and variable assignments. Let $B$ be the set of firing modes. Each object net is a tuple

$$N = (P_N, T_N, F_N^-, F_N^+)$$

where $F_N^-, F_N^+ : T_N \to (B \to (P_N \to \mathbb{N}^{\mathcal{N}}))$. Given a binding $b$ $F_N^-(t)(b)(p)$ is a multiset of object nets.

Let $P$ denote the union of all place components: $P := \bigcup_{N \in \mathcal{N}} P_N$. Assume analogously defined union sets for transitions $T$, etc.

A marking $\mu$ of an object net system maps each place to a multi-set of object nets:

$$\mu : P \to \mathbb{N}^{\mathcal{N}}$$

Here $\mu(p)(N) > 0$ describes the fact, that the place $p$ is marked with $\mu(p)(N)$ net-tokens of the type $N$.

Transitions in different nets may be synchronised via channels. In Renew channels are also used to exchange parameters. Each transition has at most one uplink, which is passive, and several downlinks, which are active in the sense that they choose the synchronisation partner. Due to this structure we obtain tree-like synchronisations. The formal definition is based on the synchronisation trees. The set of all synchronisation trees is $\mathcal{T} = \bigcup_{n \geq 0} \mathcal{T}_n$ where

$$\mathcal{T}_n := \{(t,b)[\theta_1 \cdots \theta_k] \mid t \in T \wedge b \in B \wedge \forall 1 \leq i \leq k : \theta_i \in \textstyle\bigcup_{l<n} \mathcal{T}_l\}. \tag{3}$$

The predomain $F^-$ (and analogously for $F^+$) is extended to $\widehat{F}^- : \Theta \to (P_N \to \mathbb{N}^{\mathcal{N}})$ by:

$$\widehat{F}^-((t,b)[\theta_1 \cdots \theta_k]) = F^-(t)(b) + \sum\nolimits_{i=1}^{k} \widehat{F}^-(\theta_i) \tag{4}$$

**Definition 3.1** An *Object Net System* is a tuple $OS = (\mathcal{N}, \Theta, \mu_0)$ where

- $\mathcal{N}$ is a set of object nets,
- $\Theta \subseteq \mathcal{T}$ is the set of system events, and

- $\mu_0$ is the initial marking.

As usual we have $\mu \xrightarrow{\theta} \mu'$ iff $\mu \geq \widehat{F}^-(\theta)$ and $\mu' = \mu - \widehat{F}^-(\theta) + \widehat{F}^+(\theta)$. This firing rule describes the reference semantics of object nets – for an in-deep comparison of alternative firing rules cf. [19,11].

## 3.2 Super-Dual Object Nets

Similarly to the extension of Petri nets to super-dual nets in section 2, we extend the object net formalism by using nets as pokens, called *net-pokens*. The net-pokens can be used as a dynamic *refinement* (similar to a sub routine) of the transitions they mark. Figure 7 shows a super-dual object net with nets on places and on transitions.

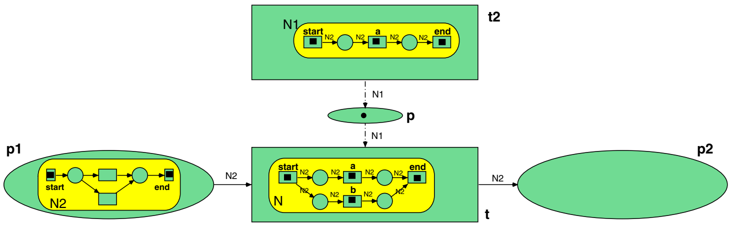


Fig. 7. A Petri net with nets as tokens for places and transitions

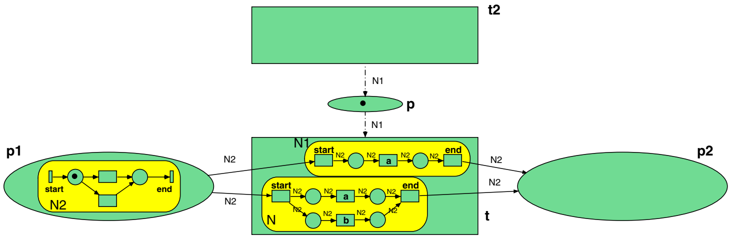


Fig. 8. Equivalent refinement after firing of place $p$

Since these refinements are defined as markings it is possible to move net-pokens using the token-game of object nets. In Figure 7 the place $p$ “fires” the net-poken from $t_2$ to $t$. Transition $t$ is then marked by two net-pokens, which means that there are two modes of refinement for $t$. The equivalent net containing the conflict between the possible refinement is given in Figure 8.

Each object net is a super-dual object net $N = (P_N, T_N, F_N^-, F_N^+, G_N^-, G_N^+)$ where $G_N^{\pm} : T_N \to (B \to (P_N \to \mathbb{N}^{\mathcal{N}}))$ define the inscriptions for glow arcs. One net models black pokens: $\blacksquare \in \mathcal{N}$. Each net-poken $N \in \mathcal{N}$ has one transition $\mathsf{start}_N$ with empty preset and one transition $\mathsf{end}_N$ with empty postset. These transitions are used to start (or end, respectively) the dynamic refinement implemented by the net-poken. From a practical point of view it is reasonable to require that the net is unmarked when the refinement is started (i.e. when transition $\mathsf{start}_N$ fires) and is unmarked again when it is ended by $\mathsf{end}_N$. We we do not adopt such a restriction here to allow a general definition.

A marking $\mu$ of a super-dual object net system maps each place and transition to a multi-set of object nets:

$$\mu : (P \cup T) \to \mathbb{N}^{\mathcal{N}}$$

In analogy to $\mathcal{T}$ we define a tree structure of places: $\mathcal{P} = \bigcup_{n \geq 0} \mathcal{P}_n$ where

$$\mathcal{P}_n = \{(p,b)[\pi_1 \cdots \pi_k] \mid p \in P \wedge b \in B \wedge \forall 1 \leq i \leq k : \pi_i \in \bigcup_{l<n} \mathcal{P}_l\}. \tag{5}$$

The mappings $\widehat{G}^-$ and $\widehat{G}^+$ are defined analogously to $\widehat{F}^-$ and $\widehat{F}^+$.

The mapping $\nu : (\mathcal{P} \cup \mathcal{T}) \to \mathbb{N}^{(P \cup T)}$ constructs a multiset by removing the nesting structure:

$$\nu(x[\xi_1 \cdots \xi_k]) := x + \sum_{i=1}^{k} \nu(\xi_i) \tag{6}$$

Here, $\nu(\theta)(t,b)$ is the number of occurences of $(t,b)$ in the nested structure $\theta$.

A dynamically refined transition $t$ is enabled in a mode $N$ where $N$ is the net-poken implementing the refinement. If $N$ is the black poken $\blacksquare$, then this transition is not refined and synchronisation is possible. If $N$ is not the black poken, it is used as a dynamic refinement of $t$. This refinement splits $t$ into a *start* and an *end* part: $(t,b,N,\mathit{start})$ and $(t,b,N,\mathit{end})$.

$$\mathcal{R} = \{(t,b,N,\alpha) \mid t \in T \wedge b \in B \wedge \blacksquare \neq N \in \mathcal{N}, \alpha \in \{\mathit{start}, \mathit{end}\}\} \tag{7}$$

**Definition 3.2** A *Super-Dual Object Net System SDOS* $= (\mathcal{N}, \Theta, \mu_0)$ consists of the following components:

- $\mathcal{N}$ is a set of object nets,
- $\Theta \subseteq (\mathcal{T} \cup \mathcal{P} \cup \mathcal{R})$ is the set of system events, and
- $\mu_0 : (P \cup T) \to \mathbb{N}^{\mathcal{N}}$ is the initial marking.

The set of system events $\Theta$ contains elements from $\mathcal{T}$, $\mathcal{P}$, and $\mathcal{R}$. So, we have different kinds of firing modes:

(i) $\theta \in \Theta \cap \mathcal{T}$: As usual we have $\mu \xrightarrow{\theta} \mu'$ iff $\forall p \in P : \mu(p) \geq \widehat{F}^-(\theta)(p)$ and $\mu(t)(\blacksquare) \geq \sum_{b \in B} \nu(\theta)(t,b)$. Then $\mu' = \mu - \widehat{F}^-(\theta) + \widehat{F}^+(\theta)$.

(ii) $\theta \in \Theta \cap \mathcal{P}$: As usual we have $\mu \xrightarrow{\pi} \mu'$ iff $\forall t \in T : \mu(t) \geq \widehat{G}^-(\pi)(t)$ and $\mu(p)(\bullet) \geq \sum_{b \in B} \nu(\theta)(p,b)$. Then $\mu' = \mu - \widehat{G}^-(\pi) + \widehat{G}^+(\pi)$.

(iii) $\theta \in \Theta \cap \mathcal{R}$: A dynamic refinement has two parts.
- The control is carried over from $t$ to a refining net-poken $N$:

$$\begin{aligned}\mu \xrightarrow{(t,b,N,\mathit{start})} \mu' \iff\ & \forall p \in P : \mu(p) \geq F^-(t)(b)(p)\\ \wedge\ & \mu(t)(N) \geq 1 \wedge \mu(\mathsf{start}_N)(\blacksquare) \geq 1\\ \wedge\ & \mu' = \mu - F^-(t)(b) + F^+(\mathsf{start}_N)(b)\end{aligned}$$

- The control is given back from $N$ to $t$:

$$\begin{aligned}\mu \xrightarrow{(t,b,N,end)} \mu' \iff\ & \forall p \in P : \mu(p) \geq F^-(\mathsf{end}_N)(b)(p)\\ \wedge\ & \mu(t)(N) \geq 1 \wedge \mu(\mathsf{end}_N)(\blacksquare) \geq 1\\ \wedge\ & \mu' = \mu - F^-(\mathsf{end}_N)(b) + F^+(t)(b)\end{aligned}$$

**Proposition 3.3** *Object nets are a special case of super-dual object nets: Each object net system is simulated by a super-dual object net system.*

**Proof.** The super-dual object net system is obtained from the object net system adding no glow arcs marking all transitions with enough black-pokens. To allow all synchronisations $\theta \in \Theta$, the transition $t$ is marked with $\max\{\nu(\theta)(t,b) \mid b \in B, \theta \in \Theta\}$ black-pokens. Then the super-dual object net system behaves the same way as the object net system since we have no refinements and no events $\theta \in \Theta \cap \mathcal{P}$. All the events $\theta \in \Theta \cap \mathcal{T}$ are enabled correspondingly and the effect is the same as for the object net system. $\square$

## 4 Simulating Super-Dual Object Nets

In our previous work [12] we have shown that super-dual nets can simulate Petri nets and, more interesting, that Petri nets can simulate super-dual nets – both in respect to the possible firing sequnces. The construction uses the dual of the $G$-component (i.e. $SD|_G^d$), renames all nodes $x$ to $x^{(d)}$ and combines it with the $F$-component. The result is the simulating Petri net $N(SD)$ (Figure 9 illustrates the construction for the net $SD$ of Figure 4).

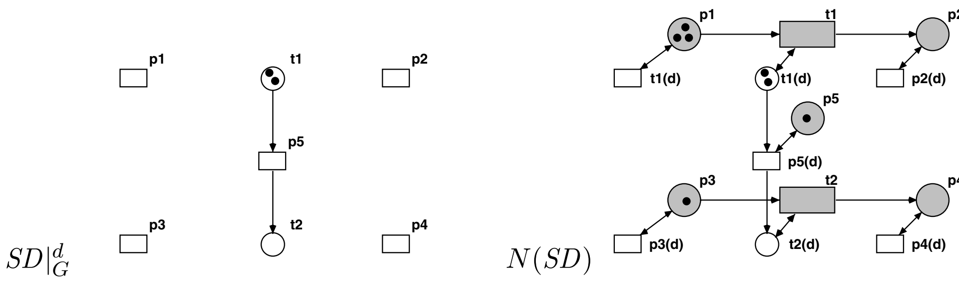


Fig. 9. The simulating net $N(SD)$

We will now lift our results to object nets and super-dual object nets to come back to the goal of this work, dynamic refinement of transitions. The construction for super-dual object nets is similar to the construction of $N(SD)$ for super-dual nets. We illustrate the construction of the simulating object net $OS(SDOS)$ at the example net from Figure 7. The construction involves two steps: In the first step the dual of the $G$-component (the unfilled nodes) is added to the $F$-component (the filled nodes) as side conditions for each object net. The refining net-pokens become net-tokens. The resulting net is given in Figure 10. Note, that the side transitions named $p_1^{(d)}$ and $p_2^{(d)}$ have no effect. The same holds for the side condition named $p$.

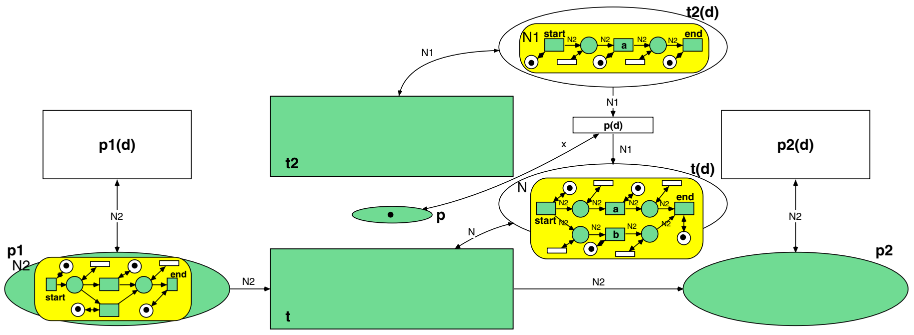


Fig. 10. Adding the dual component to the net of Figure 7

These nodes might be omitted. In the second step we split each transition $t$ of the $F$-component into two parts: $t_{start}$ and $t_{end}$, similarly to the construction suggested in the introduction (cf. Fig. 2). Transition $t_{start}$ synchronises with the input transition $\mathsf{start}_N$, i.e. it starts the refining subnet $N$. Similarly, $t_{end}$ synchronises with the output transition. The resulting net – omitting synchronisation inscriptions – is given in Figure 11.

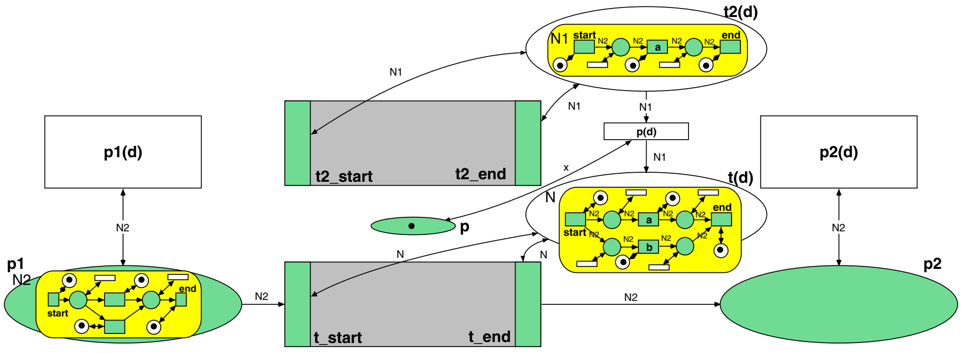


Fig. 11. Adding the start/end structure

We formalise this dualisation construction in the following. The element in the dual component corresponding to $n \in P \cup T$ is denoted $n^{(d)}$. The mapping $f^P$ maps each transition $t$ to its dual, i.e. the place $t^{(d)}$ and each place $p$ to its dual, i.e. the place $p^{(d)}$: $f^X(n) = n$ if $n \in X$ and $f^X(n) = n^{(d)}$ if $n \notin X$. The notation extends to pairs: $f^X((a,b)) = (f^X(a), f^X(b))$ and to sets: $f^X(A) = \{f^X(a) \mid a \in A\}$. This definition also extends to the nested structures $\mathcal{P}$ and $\mathcal{T}$ the usual way: $f^X(x[\xi_1 \dots \xi_k]) := f^X(x)[f^X(\xi_1) \dots f^X(\xi_k)]$.

For each marking $\mu$ in the super-dual object net the marking in the simulating net $\widetilde{\mu}$ is defined by $\widetilde{\mu}(p) = \mu(p)$ and $\widetilde{\mu}(t) = \mu(t^{(d)})$ or shorter for $n \in P \cup T$:

$$\widetilde{\mu}(n) = \mu(f^P(n)) \tag{8}$$

The simulating event $\widetilde{\theta}$ is defined according to the three kinds of firing: A synchronisation of transitions $\theta \in \Theta \cap \mathcal{T}$ is simulated by $\theta$. A synchronisation of places $\theta \in \Theta \cap \mathcal{P}$ is simulated by $\theta^{(d)}$. Both cases are subsumed by the definition $\widetilde{\theta} :=$

$f^T(\theta)$. The start event $(t, b, N, \mathit{start}) \in \Theta \cap \mathcal{R}$ is simulated by the synchronisation of $t_{start}$ (i.e. the first part of $t$) with the starting transition $\mathsf{start}_N$ of the refining net $N$:

$$(t_{start}, b)[(\mathsf{start}_N, b)[]]$$

Similarily for the event $(t, b, N, \mathit{end})$. This leads to the following definition:

$$\widetilde{\theta} := \begin{cases} f^T(\theta), & \text{if } \theta \in \Theta \cap (\mathcal{T} \cup \mathcal{P}) \\ (t_\alpha, b)[(\alpha_N, b)[]], & \text{if } \theta = (t, b, N, \alpha) \in \Theta \cap \mathcal{R}, \alpha \in \{\mathit{start}, \mathit{end}\} \end{cases} \tag{9}$$

The notation extends to sets: $\widetilde{\Theta} = \{\widetilde{\theta} \mid \theta \in \Theta\}$.

**Definition 4.1** Given a super-dual object system $\mathit{SDOS} = (\mathcal{N}, \Theta, \mu_0)$ we define the object net system

$$\mathit{OS}(\mathit{SDOS}) = (\{\widetilde{N} \mid N \in \mathcal{N}\}, \widetilde{\Theta}, \widetilde{\mu}_0)$$

where $\widetilde{N} = (f^P(P \cup T), f^T(P \cup T), \widetilde{F^-}, \widetilde{F^+})$ and with the bindings $\widetilde{B} = B \times \mathcal{N}$ the pre- and post conditions are defined by:

$$\widetilde{F^\pm}(t)(b, N)(n) = \begin{cases} F^\pm(t)(b)(p), & \text{if } n = p \in P \\ N, & \text{if } n = t^{(d)}, t \in T^{(d)} \\ \mathbf{0}, & \text{otherwise} \end{cases}$$

and

$$\widetilde{F^\pm}(p^{(d)})(b, N)(n) = \begin{cases} G^\pm(p)(b)(t), & \text{if } n = t^{(d)} \in T^{(d)} \\ N, & \text{if } n = p \in P \\ \mathbf{0}, & \text{otherwise} \end{cases}$$

Then we have the following simulation property:

**Proposition 4.2** *Let SDOS be a super-dual object net. For the object net system $\mathit{OS}(\mathit{SDOS})$ we have:*

$$\mu \xrightarrow[\mathit{SDOS}]{\theta} \mu' \iff \widetilde{\mu} \xrightarrow[\mathit{OS}(\mathit{SDOS})]{\widetilde{\theta}} \widetilde{\mu}'$$

**Proof.** It is easy to observe from the construction, that whenever an event $\theta$ of $\mathit{SDOS}$ is enabled in $\mu$ then the event $\widetilde{\theta}$ of is enabled in $\widetilde{\mu}$ in the simulation object net system $\mathit{OS}(\mathit{SDOS})$. This holds for the three cases of firing modes. No other events are enabled and the successor markings correspond. □

This approach of expressing dynamic transition refinement using object nets is implemented as a special construct in our tool RENEW. In the following we will illustrate the tool extension for the domain of dynamic workflows.

## 5 Transition Refinement: A First Approach in Renew

The concept of dynamic refinement is especially valuable in the context of workflow management systems. Dynamic refinements can be used to replace a sub workflow with a more up-to-date version respecting some preservation rules, like workflow inheritance (cf. [20]).

A first attempt to implement dynamic transition refinement in the Petri net tool Renew was done in the so-called workflow plug-in[4] [7]. Among various means for the definition and execution of workflows a so-called *task transition* was implemented. The task transition does not exactly meet our design criteria for dynamic refinement, but comes close enough to take a look.

The task transition implements two features. The first is irrelevant for the topic of this paper: The execution of a task transition may be canceled (see [7]). The second feature is close to the desired behaviour of a refined transition postulated in Section 1 (Figures 1(a) – 3).

Statically, a task transition looks like a normal transition with bold lines at the left and right side of its rectangle figure – see Figure 12. It is inscribed with a triple consisting of a task, i.e. the net[5] that refines the transition, a set of parameters to pass to this net and the expected result that should be passed back. The figure gives the semantics of a task transition in terms of a reference net. It is important to notice, that the task associated to a task transition need not be statically associated but may be exchanged at runtime. The task transition therefore puts dynamic transition refinement down to dynamic place refinement in terms of nets-within-nets.

When firing a task transition, the transition gets marked with the subnet that refines it. Renew treats this refinement token just as an ordinary token, so that the usual means for inspection and manipulation are available.

What is missing to fully meet our design criteria is – besides some implementation details – a better support for the separation of net refinement tokens from other tokens. This could be done in terms of a net type hierarchy.

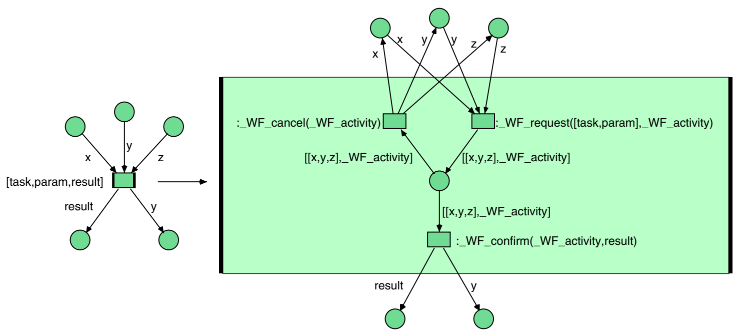


Fig. 12. The workflow task transition

[4] Renew offers a powerful plug-in concept making it easy to implement new functionality in all areas of net design, simulation and analysis.

[5] Note, that in the workflow implementation a task is not necessarily a net, but may also be Java code.

## 6 Related work

To the best of our knowledge there are no publications describing dynamic transition refinements for Petri nets. There exists, however, a small amount of publications on Petri nets that can modify their structure at runtime and, separated from these, on duality in Petri nets:

Our approach describes a special kind of Petri nets that can modify their structure via dynamic refinement of transitions. A first approach to the ability of structure changing at runtime are self-modifying nets [17] which allow for arc weights that are marking depending. A special case is the empty marking that temporarily deletes arcs from the net. Another approach to structure modification in Petri nets is that of mobile nets [1] and recursive nets [6].

Duality is an important concept in Petri's general net theory and is discussed in [5]. Petri only considers unmarked nets, so no "problems" with tokens on transitions arise. The restriction to unmarked nets is renounced by Lautenbach [15]. However, his concept of duality differs from the one presented in this paper. He considers firing in the dual reverse net $N^{rd}$. In his approach transitions become marked and places fire the tokens, which are lying on transitions, in the reversed arc direction. Additionally, contrary to our approach, for his definition a token on a transition disables its firing.

## 7 Conclusion

In this presentation we studied the dynamic refinement of transitions. Following the ideas of extending Petri nets to super-dual nets we generalised object net systems to super-dual object systems. Super-dual object net systems are nets-within-nets allowing nets as tokens both on places and on transitions. Transition marking nets, called net-pokens, may be moved around from one transition to another. They refine the transition they are actually marking. This offers the desired properties of a dynamic, run-time refinement procedure that is controlled by the net itself.

Super-dual object nets are related to an implementation of a workflow extension plugin of the Renew tool. This extension has a special notion of dynamically refinable transitions, called tasks. These task transitions are executed by instantiating a net-token that implements a sub-workflow. In accordance with our definitions these sub-workflows are the dual of normal workflows, i.e. they have a unique input transition and a unique output transition. The workflow management system can make use of this mechanism when replacing sub-workflows by updates at runtime. This can be done easily by moving net-tokens around or creating new ones at runtime, e.g. as a result of a planning process. One can think of mobile workflows implemented by mobile agents in the style of [9].

## References

[1] Busi, N., *Mobile nets*, Formal Methods for Open Object-Based Distributed Systems (1999), pp. 51–66.

[2] Design/CPN, *Design/CPN homepage*, http://www.daimi.au.dk/designCPN/ (2005).

[3] Farwer, B., *A linear logic view of object Petri nets*, Fundamenta Informaticae **37** (1999), pp. 225–246.

[4] Fehling, R., *A concept of hierarchical Petri nets with building blocks*, in: G. Rozenberg, editor, *Advances in Petri Nets 1993*, Lecture Notes in Computer Science **674** (1993), pp. 148–168.

[5] Genrich, H. J., K. Lautenbach and P. S. Thiagarajan, *Elements of general net theory*, in: Brauer, W., editor, *Net Theory and Applications, Advanced Course on General Net Theory of Processes and Systems, Hamburg, 1979*, Lecture Notes in Computer Science **84** (1980), pp. 21–163.

[6] Haddad, S. and D. Poitrenaud, *Theoretical aspects of recursive Petri nets*, in: S. Donatelli and J. Kleijn, editors, *Application and Theory of Petri Nets 1999*, Lecture Notes in Computer Science **1630** (1999), pp. 228–247.

[7] Jacob, T., O. Kummer, D. Moldt and U. Ultes-Nitsche, *Implementation of workflow systems using reference nets – security and operability aspects*, in: K. Jensen, editor, *Fourth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools* (2002).

[8] Jensen, K., "Coloured Petri nets, Basic Methods, Analysis Methods and Practical Use," EATCS monographs on theoretical computer science, Springer-Verlag, 1992.

[9] Köhler, M., D. Moldt and H. Rölke, *Modelling mobility and mobile agents using nets within nets*, in: W. v. d. Aalst and E. Best, editors, *International Conference on Application and Theory of Petri Nets 2003*, Lecture Notes in Computer Science **2679** (2003), pp. 121–140.
URL http://www.springerlink.com/link.asp?id=xf5vqh9cn0q1nukw

[10] Köhler, M. and H. Rölke, *Concurrency for mobile object-net systems*, Fundamenta Informaticae **54** (2003).

[11] Köhler, M. and H. Rölke, *Properties of Object Petri Nets*, in: J. Cortadella and W. Reisig, editors, *International Conference on Application and Theory of Petri Nets 2004*, Lecture Notes in Computer Science **3099** (2004), pp. 278–297.
URL http://www.springerlink.com/index/BLXWQ2EP7CFU2P0Y

[12] Köhler, M. and H. Rölke, *Properties super-dual nets*, Fundamenta Informaticae **72** (2006), pp. 245–254.

[13] Kummer, O., *Introduction to Petri nets and reference nets*, Sozionik-aktuell **1** (2001).
URL http://www.sozionik-aktuell.de/

[14] Kummer, O., F. Wienberg, M. Duvigneau, J. Schumacher, M. Köhler, D. Moldt, H. Rölke and R. Valk, *An extensible editor and simulation engine for Petri nets: Renew*, in: J. Cortadella and W. Reisig, editors, *International Conference on Application and Theory of Petri Nets 2004*, Lecture Notes in Computer Science **3099** (2004), pp. 484 – 493.

[15] Lautenbach, K., *Duality of marked place/transition nets*, Technical Report 18, Universität Koblenz-Landau (2003).

[16] Lomazova, I. A., *Nested Petri nets – a formalism for specification of multi-agent distributed systems*, Fundamenta Informaticae **43** (2000), pp. 195–214.

[17] Valk, R., *Self-modifying nets, a natural extension of Petri nets.*, in: Ausiello, G. and Böhm, C., editors, *Automata, Languages and Programming*, Lecture Notes in Computer Science **62** (1978), pp. 464–476.

[18] Valk, R., *Modelling concurrency by task/flow EN systems*, in: *3rd Workshop on Concurrency and Compositionality*, number 191 in GMD-Studien (1991).

[19] Valk, R., *Object Petri nets: Using the nets-within-nets paradigm*, in: J. Desel, W. Reisig and G. Rozenberg, editors, *Advanced Course on Petri Nets 2003*, Lecture Notes in Computer Science **3098** (2003), pp. 819–848.

[20] van der Aalst, W. and T. Basten, *Inheritance of workflows: An approach to tackling problems related to change*, Theoretical Computer Science **270** (2002), pp. 125–203.