A Mobility Logic for Object Net Systems

Frank Heitmann and Michael Köhler-Bußmeier

University of Hamburg, Department for Informatics
Vogt-Kölln-Straße 30, D-22527 Hamburg
{heitmann,koehler}@informatik.uni-hamburg.de

Abstract. In this paper we present work in progress on a special variant of *Object Petri Nets* and on the introduction of a *Mobility Logic* to reason about them.
The Petri nets considered in this paper allow the *vertical transport* of net tokens i.e. the transport of net tokens through different nesting levels, giving one enhanced modelling capabilities and allowing one to naturally model certain situations arising in nested structures.
The logic then allows us not only to reason about the evolution of the described system in time, but also about spatial configurations, i.e. in this logic we can express for example, that a certain object or agent is always somewhere or at a specific location. This part of our work is inspired by the work of Cardelli and Gordon on the Ambient Calculus and the Ambient Logic.

Keywords: design methods, higher-level Petri net models, nets-within-nets, mobility logic, model checking and verification

1 Introduction

Object Petri Nets are Petri Nets whose tokens may be Petri Nets again and thus may have an inner structure and activity. This approach is useful to model mobile systems and other systems arising in Computer Science which enjoy a certain nesting of structures (cf. [10] and [11]).

This approach, which is also called the *nets-within-nets* paradigm, was proposed by Valk [22, 23] for a two levelled structure and generalised in [12, 13] for arbitrary nesting structures. By now many related approaches like recursive nets [6], nested nets [18], adaptive workflow nets [19], AHO systems [9], PN^2 [8], Mobile Systems [17], and many others are known. See [14] for a detailed discussion.[1]

A variant introduced a few years ago in [15], allows the *vertical transport* of net tokens, i.e. the transport of net tokens through different nesting levels, giving one enhanced modelling capabilities and allowing one to naturally model certain situations arising in nested structures.

[1] Another line of research also dealing with nesting, but not in the field of Petri nets, is concerned with process calculi. Arguably most prominently there are the Ambient Calculus of Gordon and Cardelli [2] and the Seal Calculus [3] among many others.

Unfortunately the formalism was rather complicated and thus not well suited for neither theoretical investigations nor modelling applications. In the first part of this paper we devise a more convenient variant with regard to theoretical investigations than the variant known so far. The variant proposed here retains the ability to transport tokens in the vertical dimension, but in particular restricts the transitions participating in the firing to at most two levels.

After introducing the formalism we go on and introduce a logic that allows us not only to reason about the evolution of the described system in time, but also about spatial configurations, i.e. a logic in which we can express for example, that a certain object or agent is always somewhere or at a specific location. This part of our work is deeply inspired by the work of Cardelli and Gordon on the Ambient Calculus and the Ambient Logic [2], [1].

While the main part of this presentation deals with the introduction of the formalism and the logic and thus with definitions and examples, we also hint at work in progress regarding the complexity of certain problems for object nets and the newly developed logic. In particular we show that the reachability problem is decidable in PSpace for a specific finite-state-segment of our formalism and argue that the model checking problem for the new logic might also be in PSpace for this variant.

In the following we assume basic knowledge of Petri nets, see e.g. [20].

2 Object Nets

In [15] we presented a formalism for object nets which was rather complicated. The firing rule was particularly hard to formulate and to understand as were the events themselves. Unsurprisingly the formalism was Turing-complete, but many of the formalism's facets where not even used in the proof.

In the following we will present a stripped-down variant that still captures the essentials of the formalism in [15], namely the nets-within-nets structure, the synchronisation, and in particular the possibility to transport nets *vertically* through the channels. For an example take a look at Figure 3. An object net resides on place p' whose place p is again marked by another object net. The transitions t' and t use the same channel descriptor c and the channel properties match.[2] Ignoring the inner structure of the net tokens both transitions are activated and may fire. The successor marking is pictured in Figure 4. The net token previously on p' has travelled to p''', but it's place p is now empty, because that object net has travelled in the vertical dimension via channel c to the place p''. In the following we will give a formal description of this formalism.

An *Object Net System* (ONS for short) consist of a *system net* $\widehat{N} = (\widehat{P}, \widehat{T}, \widehat{F})$ and a finite number of *object nets* $\mathcal{N} = \{N_1, \dots, N_m\}$, $N_i = (P_i, T_i, F_i)$. Black tokens can be described by a special object net which has no places and transitions. We set $\widehat{\mathcal{N}} := \mathcal{N} \cup \widehat{N}$. Instead of P_i, T_i and so on we sometimes make use

[2] This will be defined later, for now note that the channel property $\uparrow_{N_1}$ of t means that t wants to send a object of type N_1 *upwards* and the channel property $\cap_{N_1}$ of t' means that it wants to *catch* a object of type N_1 (both via channel c).

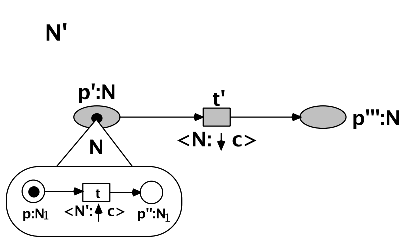

Fig. 1. Before Firing.

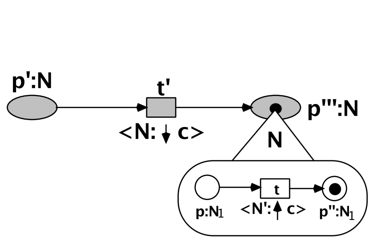

Fig. 2. After Firing.

of the notation $T(N_i) := T_i$, i.e. given an object net N the set of its transitions is denoted by $T(N)$, the set of its places by $P(N)$. We use $\mathcal{P}_\mathcal{N} := \dot{\bigcup}_{N\in\mathcal{N}} P(N)$, $\mathcal{P} := \mathcal{P}_\mathcal{N} \cup \widehat{P}$, $\mathcal{T}_\mathcal{N} := \dot{\bigcup}_{N\in\mathcal{N}} T(N)$, and $\mathcal{T} := \mathcal{T}_\mathcal{N} \cup \widehat{T}$ to denote the set of all places and transitions.

The places are all typed via the *typing function* $d : \mathcal{P} \to \mathcal{N}$. Note that no place is typed with the system net $\widehat{N}$.

Transitions are labelled with channels to allow for synchronisation. Channels consist of a *descriptor* taken from a finite set of *channel descriptors* $C_d = \{c_1, c_2, \ldots, c_n\}$ and a *channel property* $C_p = \{\Uparrow, \Downarrow, \Uparrow_{N_1}, \ldots, \Uparrow_{N_m}, \Downarrow_{N_1}, \ldots, \Downarrow_{N_m}, \cup_{N_1}, \ldots, \cup_{N_m}, \cap_{N_1}, \ldots, \cap_{N_m}\}$. A *channel* is then a element of the set $C := C_p \times C_d$, where instead of e.g. $(\Uparrow, c_1)$ we usually simply write $\Uparrow c_1$.

Since the system net is at the highest level of the hierarchy, not every channel can be used there. To ease the notation later we additionally define $\widehat{C} := (C_p \setminus \{\Uparrow, \Uparrow_{N_1}, \ldots, \Uparrow_{N_m}, \cup_{N_1}, \ldots, \cup_{N_m}\}) \times C_d$.

The *labelling functions* are now defined as

$$\widehat{l} : \widehat{T} \to (\widehat{C} \times \mathcal{N}) \cup \{\epsilon\}$$

and for each $i \in [m]$ as

$$l_i : T_i \to (C \times \widehat{\mathcal{N}}) \cup \{\epsilon\}$$

which are combined to

$$l : \mathcal{T} \to (C \times \widehat{\mathcal{N}}) \cup \{\epsilon\}$$

with $l(t) = \widehat{l}(t)$ if $t \in \widehat{T}$ and $l(t) = l_i(t)$ if $t \in T_i$.

Note that each transition is labelled with exactly one channel or ϵ. The intended meaning of $l(t) = (c, N)$ is that t synchronizes via channel c with a net of type N. In the case of $l(t) = \epsilon$ the transition t fires autonomously.

We now describe the possible labellings together with there intended meaning and the restrictions the labellings impose on the nets' structure.

1. $l(t) = \epsilon$, $t \in T(N)$. In this case there is no synchronisation and t fires in principal as in a normal p/t net.

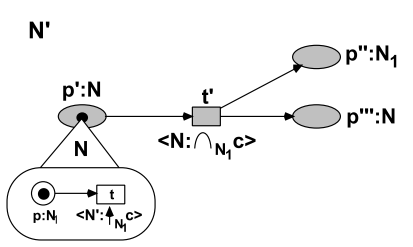

Fig. 3. Before Firing.

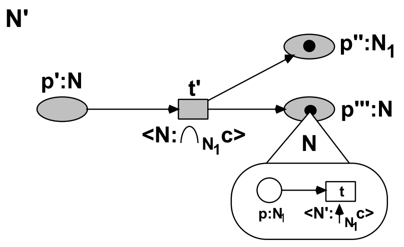

Fig. 4. After Firing.

2. $l(t) = (\Uparrow c, N')$, $t \in T(N), N \neq \widehat{N}$[3]. The labelling means that t wants to synchronize (via c) with a transition in N', where N' is a net *"above" N*, i.e. N is a net-token in N' (see Figure 1 and 2). Formally we demand a place p' in N' with $d(p') = N$ and a transition $t' \in p'^{\bullet}$ with $l(t') = (\Downarrow c, N)$.
3. $l(t') = (\Downarrow c, N)$, $t' \in T(N')$. The complement to the above case. There is now a place $p' \in {}^{\bullet}t'$ with $d(p') = N$ and in N there is a transition t with $l(t) = (\Uparrow c, N')$.
4. $l(t) = (\Uparrow_{N_1} c, N')$, $t \in T(N), N \neq \widehat{N}$. Similar to $\Uparrow$ above, t wants to synchronize (via c) with a transition in N' "above". This time additionally a net of type N_1 is send from N through c upwards to N' (resp. to a place in the postset of the transition in N' that uses the channel c). The situation is depicted in Figures 3 and 4. Note that the token on place p (Fig. 3) resp. place p'' (Fig. 4) can be an object net. Formally there is a place p' with $d(p') = N$ and a transition $t' \in p'^{\bullet}$ with $l(t') = (\cap_{N_1} c, N)$.[4] Moreover there is a place $p \in {}^{\bullet}t$ with $d(p) = N_1$ and *no place* in the postset of t of this type. In N' there is a place $p'' \in t'^{\bullet}$ with $d(p'') = N_1$ and *no place* in the preset of t' of this type. (The net of type N_1 thus travels from p (in N) to p'' (in N').)
5. $l(t') = (\cap_{N_1} c, N)$. The complement to the case above, but similar to $\Downarrow$. There is a place $p' \in {}^{\bullet}t'$ with $d(p') = N$ and in N there is a transition t with $l(t) = (\Uparrow_{N_1} c, N')$. Moreover there is a place $p \in t^{\bullet}$ with $d(p) = N_1$ and *no place* in the postset of t with this type, and also a place $p'' \in t'^{\bullet}$ in N' with $d(p'') = N_1$ and *no place* in the preset of t of this type.
6. $l(t) = (\Downarrow_{N_1} c, N')$, $t \in T(N)$. Similar to $\Downarrow$ above, t wants to synchronize via c with a transition in N' "below". This time a net of type N_1 is additionally send from N through c downwards to N' (resp. to a place in the postset of the transition in N' that uses the channel $\cup_{N_1} c$). The situation is depicted in Figures 5 and 6. Note again that the token on place p (Fig. 5) resp. place p'' (Fig. 6) can be an object net. Formally there is a place $p \in {}^{\bullet}t$ with $d(p) = N'$, a transition t' in N' with $l(t') = (\cup_{N_1} c, N)$ and moreover a place $p' \in {}^{\bullet}t$ with $d(p') = N_1$ and a place $p'' \in t'^{\bullet}$ with $d(p'') = N_1$. There is no

[3] In the system net $\widehat{N}$ the channel property $\Uparrow$ can not be used.
[4] The usage of the symbol $\cap$ shall illustrate that a net coming from below is "caught.

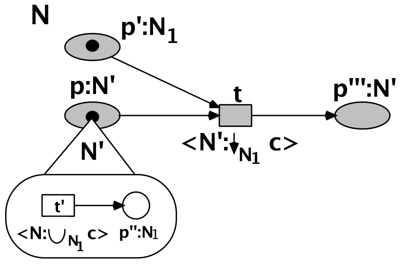

Fig. 5. Before Firing.

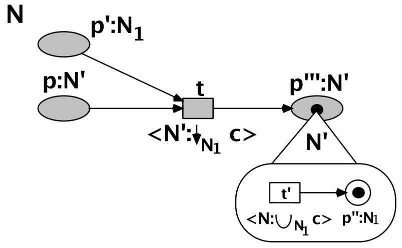

Fig. 6. After Firing.

place in the postset of t or in the preset of t' of type N_1. (The Net of type N_1 thus travels from p' (in N) to p'' (in N').)

7. $l(t') = (\cup_{N_1} c, N)$, $t' \in T(N'), N' \neq \widehat{N}$. Again the complement to the case directly above.

In addition to the above described restrictions on the nets' structure imposed by the labelling, we demand that each type appears at most once in the preset and in the postset of a transition, i.e.

$$\forall N \in \widehat{\mathcal{N}}\ \forall t \in T(N) : |\{p \mid p \in {}^\bullet t \wedge d(p) = N\}|, |\{p \mid p \in t^\bullet \wedge d(p) = N\}| \leq 1$$

Markings. To define markings, which will turn out to be nested multi-sets, let OS be an object net system as above consisting of a system net $\widehat{N}$ and a finite set of object nets $\mathcal{N}$. Furthermore let $d : \mathcal{P} \to \mathcal{N}$ be the typing function (no place is typed with the system net). Now let

$$\mathcal{M}_0(N) := \{p[\mathbf{0}] \mid p \in P(N)\}$$

for a $N \in \widehat{\mathcal{N}}$ and let $\mathcal{M}_0 := \cup_{N \in \widehat{\mathcal{N}}} \mathcal{M}_0(N)$. Note that with $\mu \in \mathcal{MS}(\mathcal{M}_0(N))$ for a fix $N \in \widehat{\mathcal{N}}$ we can describe how many *empty* net tokens reside on each place of N (this includes black tokens, which are just special net tokens in our setting). The multiset μ is thus similar to the usual multiset of places that describes a marking.

Let

$$\mathcal{M}_{i+1}(N) := \{p[\mu] \mid p \in P(N) \wedge \mu \in \mathcal{MS}(\cup_{k \leq i} \mathcal{M}_k(d(p))\} \text{ and}$$
$$\mathcal{M}_{i+1} := \cup_{N \in \widehat{\mathcal{N}}} \mathcal{M}_{i+1}(N)$$

and finally let

$$\mathcal{M} := \mathcal{MS}(\cup_{i \geq 0} \mathcal{M}_i) = \mathcal{MS}(\cup_{i \geq 0} \cup_{N \in \widehat{\mathcal{N}}} \mathcal{M}_i(N)).$$

Each *nested multiset* $\mu \in \mathcal{M}, \mu = \sum_{k=1}^{n} \widehat{p}_k[M_k]$, is a *marking* of the object net system OS, where $\widehat{p}_k$ is a place in the system net and M_k is a marking of a net-token of type $d(\widehat{p}_k)$, which again might be a nested multiset.

We extend addition, subtraction and $\leq$-relations etc. for nested multisets in the usual way, e.g. $\mu \leq \mu'$ for two nested multisets if another nested multiset ρ exists such that $\mu + \rho = \mu'$. Furthermore, we need a relation to address the nesting of markings. We write $\mu \bigtriangledown \mu'$ to indicate that the submarking μ' is contained in the marking μ within exactly one level of nesting:

$$\mu \bigtriangledown \mu' \quad \text{iff} \quad \exists p \in \mathcal{P}, \mu'' \in \mathcal{M} \,.\, \mu \equiv p[\mu'] + \mu''$$

The reflexive and transitive closure of this relation is denoted by $\bigtriangledown^*$ as usual. Thus $\mu \bigtriangledown^* \mu'$ means that μ contains μ' at some nesting level.

Note that $\mathcal{M}$ differs for different object net systems. If necessary we will denote the set of possible markings of a ONS OS by $\mathcal{M}_{OS}$, but if no ambiguities can arise, we neglect the subscript.

Given a (sub-)marking μ we use $\Pi^1(\mu)$ to abstract away the substructure of all net-tokens and $\Pi^2_N(\mu)$ for the summed up marking of all net tokens of type $N \in \mathcal{N}$ ignoring their local distribution, i.e.

$$\Pi^1(\sum_{k=1}^{n} p_k[M_k]) = \sum_{k=1}^{n} p_k$$
$$\Pi^2_N(\sum_{k=1}^{n} p_k[M_k]) = \sum_{k=1}^{n} \mathbf{1}_N(p_k) \cdot M_k,$$

where $\mathbf{1}_N : \mathcal{P} \to \{0,1\}$ with $\mathbf{1}_N(p) = 1$ iff $d(p) = N$. Note that the summation in Π^2_N is *not* recursive, i.e. a marking of a net token of type N on a deeper nesting level is not summed up (but remains in the sub-marking M_k). Defined in this way Π^2_N is useful to describe the firing rule.

Object Net Systems, Events, and the Firing Rule.

Definition 1. *An Object Net System (ONS) is a tuple $OS = (\widehat{N}, \mathcal{N}, d, l)$ with*

1. *The system net $\widehat{N}$,*
2. *a finite set of object nets $\mathcal{N}$,*
3. *the typing function $d : \mathcal{P} \to \mathcal{N}$, and*
4. *the labelling function $l : \mathcal{T} \to (C \times \widehat{\mathcal{N}}) \cup \{\epsilon\}$, which is consistent with the structural restrictions mentioned above.*

An ONS with initial marking is a tuple $OS = (\widehat{N}, \mathcal{N}, d, l, \mu_0)$ where the initial marking $\mu_0 \in \mathcal{M}$ is a marking of $\widehat{N}$, i.e. there is a k such that $\mu_0 \in \mathcal{M}_k(\widehat{N})$.

To define *events* and the *firing rule* we distinguish four cases in accordance with the labelling above:

1. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Uparrow_{N_1} c, N')$ and $l(t') = (\cap_{N_1} c, N)$ (Fig. 3 and 4).
2. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Downarrow_{N_1} c, N')$ and $l(t') = (\cup_{N_1} c, N)$ (Fig. 5 and 6).
3. $(t, t') \in \mathcal{T} \times \mathcal{T}$ with $l(t) = (\Uparrow c, N')$ and $l(t') = (\Downarrow c, N)$ (Fig. 1 and 2).
4. $t \in \mathcal{T}$ with $l(t) = \epsilon$.

The first three cases are *synchronous* events, the last one describes an *autonomous* event.

Now for the first case let μ be the current marking and let $\lambda, \lambda', \rho, \rho' \leq \mu$ be sub-markings with $\lambda' \leq \lambda$ and $\rho' \leq \rho$. The intended meaning is that λ is the sub-marking of μ enabling t' and λ' is the sub-marking (of λ) that enables t in the synchronous event. Then ρ is the resulting sub-marking with regard to t' and ρ' with regard to t. Furthermore a net of type N_1 is removed from λ' and added to ρ.

This is expressed in the firing predicate $\phi_{\Uparrow_{N_1}, \cap_{N_1}}$:

$$\begin{aligned}
\phi_{\Uparrow_{N_1}, \cap_{N_1}}(t, t', \lambda, \lambda', \rho, \rho') \iff & \\
& \Pi^1(\lambda) = \mathbf{pre}(t') \wedge \Pi^1(\rho) = \mathbf{post}(t') \wedge \\
& \Pi^1(\lambda') = \mathbf{pre}(t) \wedge \Pi^1(\rho') = \mathbf{post}(t) \wedge \\
& \forall N' \in \mathcal{N} \setminus \{N, N_1\} : \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) \wedge \\
& \forall N' \in \mathcal{N} \setminus \{N_1\} : \Pi^2_{N'}(\rho') = \Pi^2_{N'}(\lambda') \wedge \\
& \Pi^2_N(\rho) = \Pi^2_N(\lambda) - \lambda' + \rho' \wedge \\
& \Pi^2_{N_1}(\rho) = \Pi^2_{N_1}(\lambda') \wedge \\
& \Pi^2_{N_1}(\rho') = \mathbf{0}
\end{aligned} \tag{1}$$

The first two lines take care of activation of t and t' and the correct successor marking. Lines 3 and 4 handle non involved object nets and the last three lines correctly relate the different (sub-)markings with regard to the synchronous event, i.e. with regard to the two firing transitions.

The other cases are quite similar and the third and fourth case can even be seen as special (and easier) cases to the above.

For the second case the firing predicate is given by

$$\begin{aligned}
\phi_{\Downarrow_{N_1}, \cup_{N_1}}(t, t', \lambda, \lambda', \rho, \rho') \iff & \\
& \Pi^1(\lambda) = \mathbf{pre}(t) \wedge \Pi^1(\rho) = \mathbf{post}(t) \wedge \\
& \Pi^1(\lambda') = \mathbf{pre}(t') \wedge \Pi^1(\rho') = \mathbf{post}(t') \wedge \\
& \forall N' \in \mathcal{N} \setminus \{N, N_1\} : \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) \wedge \\
& \forall N' \in \mathcal{N} \setminus \{N_1\} : \Pi^2_{N'}(\rho') = \Pi^2_{N'}(\lambda') \wedge \\
& \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) - \lambda' + \rho' \wedge \\
& \Pi^2_{N_1}(\rho) = \mathbf{0} \wedge \\
& \Pi^2_{N_1}(\rho') = \Pi^2_{N_1}(\lambda)
\end{aligned} \tag{2}$$

Note that λ now enables t, λ' enables t', ρ is the resulting sub-marking with regard to t and ρ' with regard to t'. Furthermore a net of type N_1 is removed from λ and added to ρ' (and also to ρ, since $\rho' \leq \rho$). In principle the first two cases only differ in the last three lines that relate the different (sub-)markings and the firing transitions.

At last the third and fourth case:

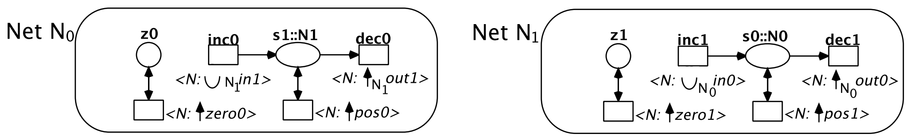

Fig. 7. The object nets N_0 and N_1 (from Theorem 1).

$$\begin{aligned}\phi_{\Uparrow,\Downarrow}(t,t',\lambda,\lambda',\rho,\rho') \iff & \\ & \Pi^1(\lambda) = \mathbf{pre}(t') \wedge \Pi^1(\rho) = \mathbf{post}(t') \wedge \\ & \Pi^1(\lambda') = \mathbf{pre}(t) \wedge \Pi^1(\rho') = \mathbf{post}(t) \wedge \\ & \forall N \in \mathcal{N} \setminus \{N'\} : \Pi^2_N(\rho) = \Pi^2_N(\lambda) \wedge \\ & \Pi^2_{N'}(\rho) = \Pi^2_{N'}(\lambda) - \lambda' + \rho'\end{aligned} \tag{3}$$

$$\begin{aligned}\phi_{\epsilon}(t,\lambda,\rho) \iff & \\ & \Pi^1(\lambda) = \mathbf{pre}(t) \wedge \Pi^1(\rho) = \mathbf{post}(t) \wedge \\ & \forall N \in \mathcal{N} : \Pi^2_N(\rho) = \Pi^2_N(\lambda)\end{aligned} \tag{4}$$

Note that the four firing predicates might a first glance look cumbersome, but are quite similar and in particular restrict every firing to two levels, which is far better tractable from a theoretical point of view than the firing rule introduced in [15] where a tree of synchronous transitions was able to fire. The firing rule can now be stated as follows:

Definition 2 (Firing Rule). *Let OS be an* ONS *and* $\mu, \mu' \in \mathcal{M}$ *markings. The synchronous event* (t, t') *is* enabled in μ for the mode $(\lambda, \lambda', \rho, \rho') \in \mathcal{M}^4$ *iff* $\lambda' \leq \lambda \leq \mu$, $\rho' \leq \rho$ *and one of* $\phi_{\Uparrow_{N_1},\cap_{N_1}}$, $\phi_{\Downarrow_{N_1},\cup_{N_1}}$, *or* $\phi_{\Uparrow,\Downarrow}$ *holds for* $(t, t', \lambda, \lambda', \rho, \rho')$, *according to the labelling of* t *and* t'.

An autonomous event $t, l(t) = \epsilon$ *is enabled in* μ *for the mode* (λ, ρ) *iff* $\lambda \leq \mu$ *and* ϕ_ϵ *holds.*

An event ϑ *that is enabled in* μ *for a mode can fire:* $\mu \xrightarrow[OS]{\vartheta} \mu'$. *The resulting successor marking is defined as* $\mu' = \mu - \lambda + \rho$.

The set of events is denoted by Θ. *Firing is extend to sequences* $w \in \Theta*$ *in the usual way. The set of reachable markings from a marking* μ *is denoted by* $RS_{OS}(\mu)$ *or simply* $RS(\mu)$. *The reachability problem asks given an* ONS *OS with initial marking* μ_0 *and a marking* μ, *if* $\mu \in RS_{OS}(\mu_0)$ *holds.*

2.1 Turing-Completeness of Object Net Systems

In [15] we have shown that the there defined object net formalism can directly simulate counter programs and thus is Turing-complete. We have severely restricted the formalism here, but retained the general ability to transfer net-tokens in the vertical dimension of the nested marking. The formalism devised here remains Turing-complete and indeed the proof in [15] can be easily adjusted to our new setting. We will only sketch the proof here.

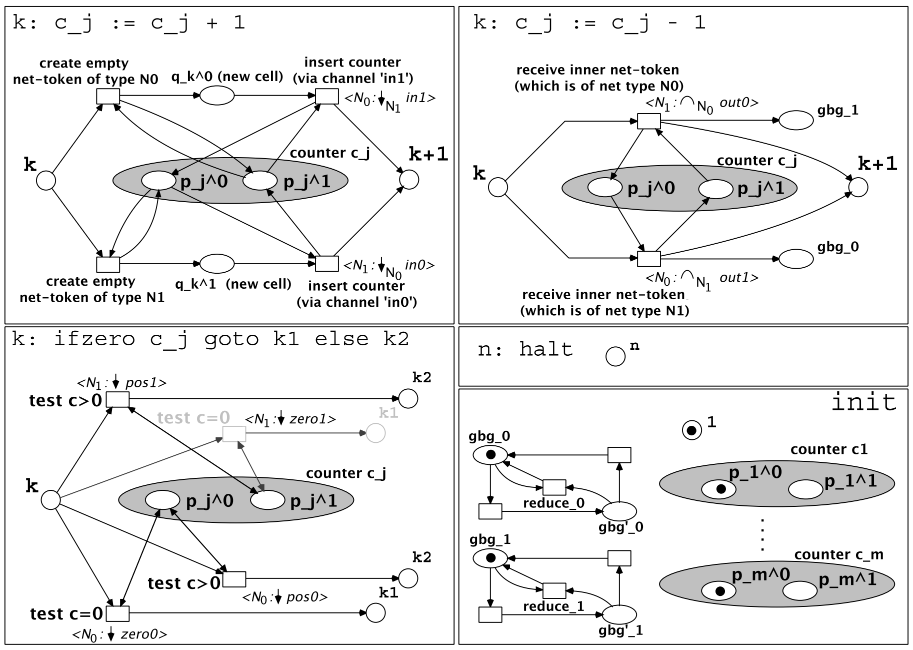

Fig. 8. Net fragments for the simulation of counter programs (from Theorem 1).

Theorem 1. *Object net systems can directly simulate counter programs and thus the reachability problem is undecidable for them.*

Proof sketch. In counter programs one has a fixed number of counters, an increase and a decrease operation and an operation that tests if a certain counter is zero and jumps accordingly.

The counters are encoded by the nesting depth of the two object nets depicted in Figure 7. For a counter c_j two places p_j^0 and p_j^1 will exist in the system net, where p_j^0 is typed with N_0 and p_j^1 with N_1. Initially each place p_j^0 will hold a object net of type N_0 whose place z_0 will be marked by a black token. If the counter is increased a net token of type N_1 is created and the aforementioned net token will be put *into* it on place s_0. The net-tokens are then either packed into each other or unpacked from each other depending on the increase or decrease operation used in the counter program.

Figure 8 shows the net fragments for each of the possible counter program commands. Note how in the increase operation either a net token of type N_0 or of type N_1 is created, depending on the most outer net token currently encoding the state of the counter. The current net-token is then put into the new one by use of the channel in_0 or in_1 again depending on the current state of the counter. If in the current state of the counter the place p_j^0 is marked, then the channel in_0 is used by synchronizing with the identically named channel in the net of type N_1 residing on q^1_k. The net of type N_0 is taken from p^0_j transported down via the channel in_0 to the place s_0 in the net of type N_1. This very net then ends up at place p^1_j representing the new state of the counter. Note how the channel properties of the channel in_0 match. Unpacking for the decrease operation is encoded similarly. The zero test can be easily encoded by trying to synchronise with the transition in N_0 which uses channel $zero_0$. Only the net on the lowest level has the place z_0 marked. Garbage collection (on the lower right of Figure 8) is only needed to make final markings unique and is not discussed here.

Details on the construction can be found in [15]. $\diamond$

Safeness for Object Net Systems. Introducing safeness for ONS and thus restricting the state space to a finite size is part of current work and we only want to touch the topic and not go into details.

In general it is a good idea to restrict the size of the state space by introducing safeness for a given Petri net formalism. Unfortunately due to the nesting depth the state space might non the less become very big for ONS. It seems that the reachability problem is far beyond PSpace (for 1-safe p/t nets and also for safe Eos, a nets-within-nets formalism with only two levels, reachability is PSpace-complete [5], [16]) and indeed seems to be Exptime-complete.

Definition 3 (Safeness). *Let OS be an* ONS *with initial marking μ_0. OS is* safe *iff $|RS_{OS}(\mu_0)| < \infty$, i.e. if the set of reachable markings of OS is finite.*

OS is strongly safe *iff OS is safe and each net-token is 1-safe, that is, if $|RS_{OS}(\mu_0)| < \infty$ and $\forall \mu \in RS(\mu_0)\ \forall \mu' \mathrel{\triangle}^* \mu\ \forall p \in \mathcal{P} : \Pi^1(\mu')(p) \leq 1$*

Conjecture 1. The Reachability problem for safe ONS is Exptime-complete.

Additionally forbidding the creation of net-tokens on the other hand gives us the opportunity to solve the reachability problem in PSpace again. This restriction is indeed not as severe as it might seem at first glance, because it simply does not allow the creation or destruction of net-tokens which - if net-tokens are interpreted as agents - might not be so undesirable at all.

Definition 4. *Let OS be an* ONS *and $\mu = \sum_{k=1}^{n} p_k[M_k]$ be a marking of OS. With $\Pi^3_N(\mu)$ we denote the* number *of net-tokens of type N present in μ, i.e.*

$$\Pi^3_N(\sum_{k=1}^{n} p_k[M_k]) = \sum_{k=1}^{n} \mathbf{1}_N(p_k) + \Pi^3_N(M_k).$$

Note that $\Pi^3_N(\mu)$ is calculated recursively.

Theorem 2. *Let OS be a strongly safe* ONS *in which no object nets are created nor destroyed, i.e. if $\mu, \mu' \in RS_{OS}(\mu_0)$ and μ' is an immediate successor of μ, then $\Pi^3_N(\mu) = \Pi^3_N(\mu')$ for all $N \in \mathcal{N}$.*[5] *Then the reachability problem is solvable in* PSpace.

[5] The firing rule ensures that the object nets are actually the "same nets.

Proof sketch. Assume that k net-tokens are present in OS. Furthermore for $N \in \widehat{\mathcal{N}}$ let P_N be the set of places of N. Let $n = max\{P_N \mid N \in \widehat{\mathcal{N}}\}$ be the maximal number of places of the involved nets.

We give an (rough) upper bound for the number of reachable markings. Assume that all net-tokens reside on one system net place $\widehat{p}$. Ignoring the nesting and in particular the structure of nested tokens and thus only taking into account if a place of a net-token is marked or not, we have an upper bound of $(2^n)^k = 2^{n \cdot k}$ different markings, because each net-token is 1-safe and thus has at most 2^n different markings and because we have k net-tokens.

To give a bound for the number of different nestings, we use Cayley's formula according to which the number of different trees on n nodes is n^{n-2} [4]. Note that the nesting of the k net-tokens can be represented by forests, i.e. by a set of trees. The root of each tree represents a net-token residing on $\widehat{p}$. The children of a node v of the tree represent net-tokens residing in the net-token represented by v.

At most we have k trees and each of the k net-tokens may be part of one of those trees, so we have at most k^k different trees. In each of these possibilities we have at most k trees and each tree has at most k nodes, so we have an upper bound of $k^k \cdot k \cdot k^{k-2} < k^{2k}$ for the number of forests on k nodes (where the last factor comes from Cayley's formula).[6]

Taking the number $m := |\widehat{P}|$ of system net places into account we end up with at most $(k^{2k} \cdot 2^{nk})^m \leq (k^{2k} \cdot 2^{nk})^n = k^{2kn} \cdot 2^{nkn} = 2^{\log k \cdot 2kn} \cdot 2^{nkn} < 2^{2kkn+nkn}$

Now note that $2kkn + nkn$ is a polynomial in the input length and thus the technique Savitch used to prove that PSpace and NPSpace are equal (cf. [21]) is applicable. Since we can furthermore test in polynomial space if a marking is reachable from another marking and also if a marking is identical to another all necessary operations are possible in polynomial space in the input length, and thus the reachability problem is solvable in polynomial space.[7] $\diamond$

The result above can be easily complemented by a proof of PSpace-hardness. Indeed the proof in [7] that the reachability problem is PSpace-hard for ppGSMs can be carried over one-to-one to the setting above.

3 A Mobility Logic for Object Nets

In common logics used in formal verification like CTL and LTL statements about time are possible, e.g. it is possible to ask if a certain state is *ever* reached or if all states reached (in time) have a certain property. In the context of modelling formalisms which allow to model the local distribution of certain objects a logic which also takes locality into account is highly useful. One then might ask questions like e.g. if a certain object will be at a certain position at a certain point in time, or if a certain object will at least be somewhere.

[6] This bound is only a rough approximation, but it suffices here.

[7] For a more detailed discussion of this technique we direct the reader to [7], where we also used it to prove that polynomial space suffices to decide reachability for ppGSMs.

A prominent example for this is the Ambient Calculus and the Ambient Logic associated with it [2], [1], by which our work is deeply inspired. The ambient calculus can be used to describe processes which do not only evolve in time, but also in space. The ambient logic can then be used to express properties of such processes taking into account both, time *and* space.

For the object net formalism presented above a similar logic is desirable. In the example above one could then for example ask the question if it is possible for an agent to enter a certain vehicle. In the following we devise a *Mobility Logic for Object Net Systems* in which satisfaction of formulas will be defined with regard to a given marking of a given object net system, i.e. $OS, \mu \models \mathcal{F}$ holds that the marking μ of the object net system OS satisfies the closed formula $\mathcal{F}$. We usually omit the ONS OS.

The satisfaction relation $\equiv$ is based on the *structural congruence relation.* Intuitively, this relation equalizes markings up to 'commutativity' and 'associativity' of submarkings.

$$\begin{array}{l}
\mu \equiv \mu \\
\mu \equiv \mu' \Rightarrow \mu' \equiv \mu \\
\mu \equiv \mu', \mu' \equiv \mu'' \Rightarrow \mu \equiv \mu'' \\
\\
\mu + \mu' \equiv \mu' + \mu \\
(\mu + \mu') + \mu'' \equiv \mu + (\mu' + \mu'') \\
\mu + \mathbf{0} \equiv \mu
\end{array}$$

Formulas are defined inductively by the following grammar:

$$\begin{array}{rl}
\phi := & \mathbf{T} \mid \neg\phi \mid (\phi \vee \phi) \mid \\
& \mathbf{0} \mid \ p[\phi] \mid (\phi + \phi) \mid \\
& \Diamond\phi \mid \ \boxdot\, \phi
\end{array}$$

To define the semantic, let OS be an ONS and $\mathcal{M}$ be the set of markings of OS. The truth of a formula ϕ as defined above is then given by the recursively defined relation $\models$ with regard to OS. Note that we used $\mu \bigtriangledown \mu'$ to indicate that the submarking μ' is contained in the marking μ within exactly one level of nesting and that $\mu \bigtriangledown^* \mu'$ means that μ contains μ' at some nesting level.

$$\begin{array}{lll}
\forall \mu \in \mathcal{M}\ \mu \models T & & \\
\forall \mu \in \mathcal{M}\ \mu \models \neg\phi & \text{iff} & \mu \not\models \phi \\
\forall \mu \in \mathcal{M}\ \mu \models (\phi_1 \vee \phi_2) & \text{iff} & \mu \models \phi_1 \text{ or } \mu \models \phi_2 \\
\forall \mu \in \mathcal{M}\ \mu \models \mathbf{0} & \text{iff} & \mu \equiv \mathbf{0} \\
\forall \mu \in \mathcal{M}\ \mu \models p[\phi] & \text{iff} & \exists \mu' \in \mathcal{M}. \mu \equiv p[\mu'] \wedge \mu' \models \phi \\
\forall \mu \in \mathcal{M}\ \mu \models (\phi_1 + \phi_2) & \text{iff} & \exists \mu', \mu'' \in \mathcal{M}. \mu \equiv \mu' + \mu'' \wedge \\
& & \quad \mu' \models \phi_1 \wedge \mu'' \models \phi_2 \\
\forall \mu \in \mathcal{M}\ \mu \models \Diamond\phi & \text{iff} & \exists \mu' \in \mathcal{M}. \mu \xrightarrow{*} \mu' \wedge \mu' \models \phi \\
\forall \mu \in \mathcal{M}\ \mu \models \boxdot\phi & \text{iff} & \exists \mu' \in \mathcal{M}. \mu \bigtriangledown^* \mu' \wedge \mu' \models \phi
\end{array}$$

Example 1. We give a few examples for formulas of the mobility logic:

- $\mu \models (p[T] + T)$ is true, if the place p is marked in μ at nesting depth 0, that is μ is congruent to $p[\mu'] + \mu''$, where μ' and μ'' are submarkings.
- $\mu \models \Diamond(p_1[\mathbf{0}] + p_2[p[\mathbf{0}]])$ is true, if the place p_1 is marked in μ with an empty net-token (which might also symbolize a black token) and p_2 is marked with an object net whose place p is marked by an empty net-token (or a black token). In this way the standard reachability problem can be formulated.
- $\mu \models \boxdot p[T]$ is true if in the marking μ a object net N resides (in some nesting depth) whose place p is marked (in an arbitrary way). Note that no other place of N might be marked. To allow this one would use the formula $\boxdot(p[T] + T)$.
- $\mu \models \Diamond \boxdot p[T]$ is true if from μ a marking μ' is reachable that satisfies $\boxdot p[T]$ (see above).

Model Checking the Mobility Logic against ONS. Given an ONS OS, a marking μ of OS, and a formula ϕ of the mobility logic, we want to decide if $OS, \mu \models \phi$ holds, i.e. if ϕ is satisfied in the marking μ of OS.

Since we can easily express reachability with the operator $\Diamond$ in the logic, the problem is undecidable for the general ONS-formalism due to Theorem 1 above.

For the restricted ONS-formalisms which enjoy a finite state space, i.e. for safe ONS, strongly safe ONS, and for strongly safe ONS, with a constant number of net-tokens, the question is currently open. We suspect that in the last case PSpace again suffices and that we need exponential time or exponential space in the first two cases.

Note that to decide if $\mu \models \boxdot p[T] + T$ holds, it is sufficient to scan over μ in linear time and test if p is present somewhere - at least if μ is given as a string as in our examples. This test should thus be easy to do in a subroutine in a PSpace-algorithm.

The nesting of operators on the other hand, might complicate things since to test $\mu \models \Diamond \boxdot \Diamond \phi$ the formula $\Diamond \phi$ might be true for an object net somewhere which is reached sometime, but only if this object net is then treated in isolation (and not taking the other nets into account). So we might have to start subroutines with different object net systems to decide if certain sub-formulas hold or not.

4 Conclusion and Outlook

The formalisms introduced in this paper, the considered problems, and the results and conjectures so far are summarized in Table 1 below. The entries with a question mark are conjectures.

We have introduced a simplified version of the object net formalism from [15], which still allows the transportation of net-tokens in the vertical dimension, but which has a much easier firing rule, in particular restricting the transitions participating in the firing to at most two levels of nesting.

Table 1. The results and conjectures so far.

	Reachability	Model Checking Mob. Log.
ONS	undecidable	undecidable
safe ONS	EXPTIME-complete ?	EXPTIME-complete ?
strongly safe ONS	EXPTIME-complete ?	EXPTIME-complete ?
strongly safe ONS with a constant number of net-tokens	PSPACE-complete	PSPACE-complete ?

We have shown that the formalism remains Turing-complete and have introduced several restrictions of the formalism to a finite state space: safe ONS, strongly safe ONS, and strongly safe ONS with a constant number of net-tokens. For the last formalism we have shown that the reachability problem is solvable in polynomial space, but all these formalisms deserve a more thorough investigation in the future.

We have then introduced a mobility logic for object net systems which allows to reason about the nesting and thus about the location of net-tokens. In future work we want to focus on the model checking problem for this logic and variants of the object net formalism. We also want to investigate variants of the logic where reasonable. For example it might be interesting to restrict the allowed nesting of operators to prevent e.g. formulas of the form $\Diamond \boxdot \Diamond\phi$.

References

1. Cardelli, L., Gordon, A.D.: Anytime, anywhere. modal logics for mobile ambients. In: Proceedings of the 27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. pp. 365–377. ACM Press (2000)
2. Cardelli, L., Gordon, A.D.: Mobile ambients. Theoretical Computer Science 240, 177–213 (2000)
3. Castagna, G., Vitek, J., Nardelli, F.Z.: The seal calculus. Information and Computation 201, 1–54 (2005)
4. Cayley, A.: A theorem on trees. Quarterly Journal of Pure and Applied Mathematics 23, 376–378 (1889)
5. Esparza, J.: Decidability and complexity of petri net problems – an introduction. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, Advances in Petri Nets. Lecture Notes in Computer Science, vol. 1491, pp. 374–428. Springer-Verlag (1998)
6. Haddad, S., Poitrenaud, D.: Theoretical aspects of recursive Petri nets. In: Donatelli, S., Kleijn, J. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 1639, pp. 228–247. Springer-Verlag (1999)
7. Heitmann, F., Köhler-Bußmeier, M.: P- and t-systems in the nets-within-nets-formalism. In: Pomello, L., Haddad, S. (eds.) To Appear in 33rd International Conference on Application and Theory of Petri Nets and Concurrency. Lecture Notes in Computer Science, Springer-Verlag (2012)
8. Hiraishi, K.: PN^2: An elementary model for design and analysis of multi-agent systems. In: Arbab, F., Talcott, C.L. (eds.) Coordination Models and Languages, COORDINATION 2002. Lecture Notes in Computer Science, vol. 2315, pp. 220–235. Springer-Verlag (2002)

9. Hoffmann, K., Ehrig, H., Mossakowski, T.: High-level nets with nets and rules as tokens. In: Application and Theory of Petri Nets and Other Models of Concurrency. Lecture Notes in Computer Science, vol. 3536, pp. 268 – 288. Springer-Verlag (2005)
10. Köhler, M., Moldt, D., Rölke, H.: Modeling the behaviour of Petri net agents. In: Colom, J.M., Koutny, M. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 2075, pp. 224–241. Springer-Verlag (2001)
11. Köhler, M., Moldt, D., Rölke, H.: Modelling mobility and mobile agents using nets within nets. In: v. d. Aalst, W., Best, E. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 2679, pp. 121–140. Springer-Verlag (2003)
12. Köhler, M., Rölke, H.: Concurrency for mobile object-net systems. Fundamenta Informaticae 54(2-3) (2003)
13. Köhler, M., Rölke, H.: Properties of Object Petri Nets. In: Cortadella, J., Reisig, W. (eds.) Application and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 3099, pp. 278–297. Springer-Verlag (2004)
14. Köhler-Bußmeier, M.: A survey of elementary object systems: Decidability results. Report of the Department of Informatics, Universität Hamburg (2011)
15. Köhler-Bußmeier, M., Heitmann, F.: On the expressiveness of communication channels for object nets. Fundamenta Informaticae 93(1-3), 205–219 (2009)
16. Köhler-Bußmeier, M., Heitmann, F.: Safeness for object nets. Fundamenta Informaticae 101(1-2), 29–43 (2010)
17. Lakos, C.: A Petri net view of mobility. In: Formal Techniques for Networked and Distributed Systems (FORTE 2005). Lecture Notes in Computer Science, vol. 3731, pp. 174–188. Springer-Verlag (2005)
18. Lomazova, I.A.: Nested Petri nets – a formalism for specification of multi-agent distributed systems. Fundamenta Informaticae 43(1-4), 195–214 (2000)
19. Lomazova, I.A., van Hee, K.M., Oanea, O., Serebrenik, A., Sidorova, N., Voorhoeve, M.: Nested nets for adaptive systems. In: Application and Theory of Petri Nets and Other Models of Concurrency. pp. 241–260. Lecture Notes in Computer Science, Springer-Verlag (2006)
20. Reisig, W., Rozenberg, G. (eds.): Lectures on Petri Nets I: Basic Models, Lecture Notes in Computer Science, vol. 1491. Springer-Verlag (1998)
21. Savitch, W.: Relationship between nondeterministic and deterministic tape complexities. J. on Computer and System Sciences 4, 177–192 (1970)
22. Valk, R.: Modelling concurrency by task/flow EN systems. In: 3rd Workshop on Concurrency and Compositionality. No. 191 in GMD-Studien, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn (1991)
23. Valk, R.: Object Petri nets: Using the nets-within-nets paradigm. In: Desel, J., Reisig, W., Rozenberg, G. (eds.) Advanced Course on Petri Nets 2003. Lecture Notes in Computer Science, vol. 3098, pp. 819–848. Springer-Verlag (2003)