Object Petri Nets: Definitions, Properties, and Related Models

Michael Köhler

University of Hamburg, Department of Computer Science
Vogt-Kölln-Str. 30, D-22527 Hamburg
koehler@informatik.uni-hamburg.de

Abstract. In this presentation the decidability issues of formalisms that allow Petri nets as tokens are studied. Especially for "elementary object-net systems" defined by Valk the decidability of the reachability and the boundedness problem is considered. It is shown that reachability becomes undecidable while boundedness remains decidable for elementary object-net systems. Furthermore it is shown that even for minimal extensions the formalism obtains the power of Turing machines.

Keywords: decidability, nets within nets, object Petri nets

1 Introduction

In this presentation decidability issues for different kinds of Object Petri Nets are studied. Object Petri Net formalisms use complex objects (defined in some object-oriented specification language) as tokens. The *"nets within nets"* approach of Valk assumes that these objects are Petri nets again: In [Val98] "unary elementary object systems" (Ueons) are introduced – a basic model restricted to a two-level hierarchy. In the following the term "object-net" is used for the "nets within nets" interpretation. Related models of this "nets within nets" approach are Nested Petri Nets [LS00], Linear Logic Object Petri Nets [Far99], Reference Nets [Kum02] and Mobile Object-Net Systems [KR03]. Furthermore, there is a close connection to mobility calculi, like the ambient calculus [CGG99].

The main question is whether results for Petri nets carry over for object-nets. The analysis of decidable properties of "nets within nets" formalisms gives a deeper insight in the question whether they are just a more convenient representation of another – possibly larger – Petri net model, or whether they are a real extension with more computational power – similar to Self-Modifying Nets [Val78] which are a real extension.

The paper is structured as follows. Section 2 gives an informal introduction into the *nets within nets* approach. Section 3 recalls the formal definition of "unary elementary object systems" together with two variants of the firing rule, namely reference and value semantics. Some basic properties concerning the relationship of the two semantics or the existence of linear invariants are proven. Section 4 defines the generalised model of Object-Nets which allow an arbitrary deep nesting structure. The relationship towards other nested models is studied. Section 5 analyses the decidability of the reachability and boundedness problem for Ueons. Furthermore, it is proven, that the generalised formalism has the power of Turing machines. The work closes with a conclusion.

2 Object-oriented Petri Nets and Nets within Nets

The paradigm of nets within nets due to Valk [Val98] formalises the aspect that tokens of a Petri net can be nets again. Taking this point of view it is possible to model e.g. mobility very naturally: A mobile entity is described by a Petri net which is a token of another Petri net describing the whole surrounding system (cf. [KMR03]).

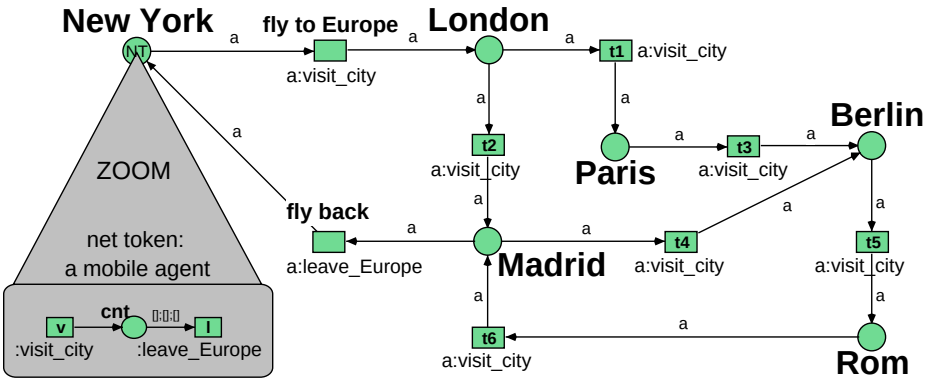

Fig. 1. A Mobile Agent as a Net Token

To give an example we consider a situation in Figure 1 where we have a two-level hierarchy. The net token is then called the "object-net", the surrounding net is called the "system-net". An intuitive interpretation of this model is a scenario, where each object-net models a mobile agent and the system-net models the agent system. In this example the agent a wants to travel from New York to Europe and he does not come back until he has visited at least three european cities. Initially the agent is a net token on the place New York – indicated by the ZOOM. The places in the system-net describes the cities of the scenario, the transitions movements between them.

Object and system-nets synchronise via *channels*. The channels are denoted as transition inscriptions of the form a:visit_city in the system-net and :visit_city in the object-net. The asymmetry is due to the fact, that the object-net (the agent a) is known in the system-net but not vice versa. Transitions with corresponding channels must fire synchronously. Transitions without an inscription can fire autonomously and concurrently to other enabled transitions.

Let us look at an example process of the system in Figure 1: The first firing step is a synchronous firing of the transitions fly to Europe and the transition v ("visit") wrt. the channel visit_city. As a result one black token is generated on the place cnt (the counter) inside the object-net and the whole object-net token is located in London. Then the agent moves to Madrid, generating a second token on the place cnt. This time the agent cannot fly back to New York, since he has only two tokens on the place cnt, while three are needed to activate the channel leave_Europe. So, the transitions fly back and l ("leave") are not activated. The

sole possibility is to travel to Berlin, to Rom, and afterwards to Madrid again. Now, the agent can fly back, since he has five tokens on the place cnt.

3 Unary Elementary Object Systems

The simplest model of "nets within nets" are "Unary Elementary Object Systems" (UEOS) defined by [Val98]. They are called "elementary" since the nesting hierarchy is limited to a depth of two. On the top level there is one so called *system-net* which has instance of one – therefore the term unary – *object-net*. Here, we give a generalised version of UEOS, since Place/ Transition nets (short: P/T-nets) are considered (instead of EN systems as in [Val98]).

In the following we use the notation $\mathbf{x}$ for elements of the whole object-net system, $\hat{x}$ for elements of the object-net, and x for elements of the system-net.

Definition 1. *An* unary elementary object system *is a tuple* $OS = (N, \hat{N}, \rho)$, *such that:*

- *The* system-net $N = (P, T, pre, post, M_0)$ *is a P/T-net with* $|M_0| = 1$ *and* $|{}^\bullet t|, |t^\bullet| > 0$ *for all* $t \in T$.
- *The* object-net $\hat{N} = (\hat{P}, \hat{T}, \hat{pre}, \hat{post}, \hat{M}_0)$ *is a P/T-net disjoint from the system-net:* $(P \cup T) \cap (\hat{P} \cup \hat{T}) = \emptyset$.
- $\rho \subseteq T \times \hat{T}$ *is the* interaction relation.

The set of synchronising transitions in the system-net is $T_\rho := (\cdot\rho\hat{T})$. The set of synchronisation free transitions is $T_{\bar{\rho}} := T \setminus T_\rho$. Analogously, the set of synchronising transitions in the object-net is $\hat{T}_\rho := (T\rho\cdot)$. The set of synchronisation free transitions is $\hat{T}_{\bar{\rho}} := \hat{T} \setminus \hat{T}_\rho$. Thus, the set of transitions is

$$\mathbf{T} := (T_{\bar{\rho}} \cup \hat{T}_{\bar{\rho}} \cup \rho)$$

.

3.1 Locality and Distribution

Nets within nets can be investigated wrt. the so called *reference* and *value semantics* – similarly to "call by reference" and "call by value" in programming languages. For reference semantics identifiers are used as references to the net tokens. The same reference can be used as a token for more than one place. For value semantics each net token is modelled as a different object, i.e. each net token lies on its specific place.

The main difference between value and reference semantic is due to the handling of distribution. Reference semantics assumes a global name space and thus considers the sum $\Pi^2(\mathbf{M})$.

In Figure 2 a firing sequence wrt. reference semantics is illustrated. First, the firing of transition t_1 creates two references on the places s_2 and s_3. The concurrent firing of (t_2, t_{11}) and (t_3, t_{12}) moves the references to s_2 and s_3 resp. and creates the marking $s_{13} + s_{14}$ in the object-net. So, the transition t_{13} of the object-net is activated and fires $s_{13} + s_{14}$ to s_{15}.

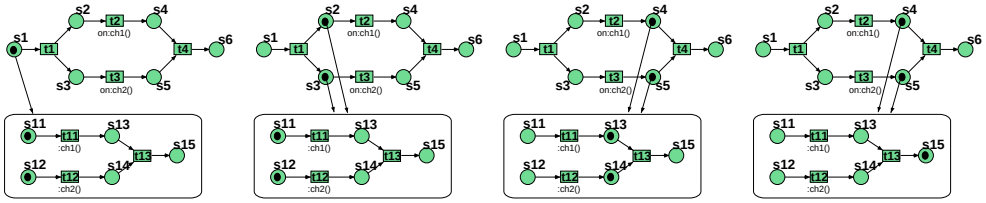

Fig. 2. Firing Sequence wrt. Reference Semantics

It is easy to see, that this sequence contradicts our intuition to deal with *two* independent net tokens. Value semantics provides the intended behaviour (cf. Fig. 3): The firing of t_1 created two net tokens, which are copies of the original one. The marking of the net token on p_1 is distributed to the copies. The distribution chosen in this sequence allows the two synchronisations (t_2, t_{11}) and (t_3, t_{12}). The effect of firing (t_2, t_{11}) modifies the marking of the net token on place s_2, but not the copy on s_3 – similarly for (t_3, t_{12}). Since the tokens on s_{13} und s_{14} are in different net tokens, transitions t_{13} is *not* enabled. The two net copies have to be recombined by the system-net transition t_4, resulting in a net token with the marking $s_{13} + s_{14}$.

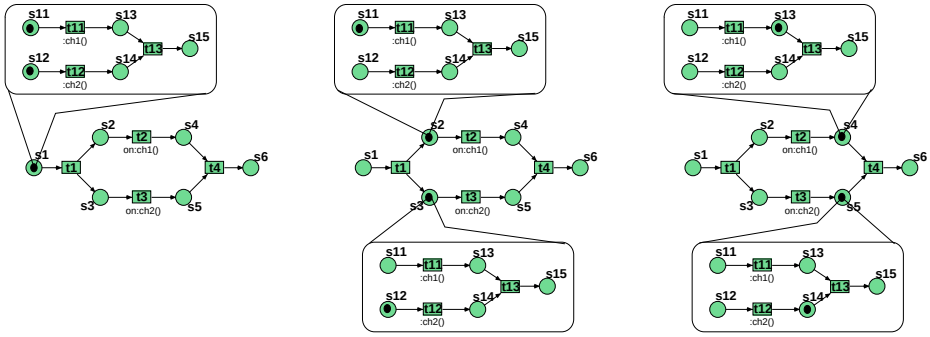

Fig. 3. Firing Sequence wrt. Value Semantics

It can be concluded, that value semantics is more intuitive when dealing with mobile objects in a distributed system, since for value semantics each place p denotes a location for the net tokens independent from all other locations.

3.2 Reference Semantics

For reference semantics net token are interpreted as references, like pointers in programming languages.

Definition 2. *A* marking *wrt. reference semantics of OS is a multiset* $\mathbf{M} \in \mathcal{M}_r := MS(P \cup \hat{P})$. *The* initial marking *of OS wrt. reference semantics is* $\mathbf{M}_0 = M_0 + \hat{M}_0$.

The firing rule is defined for three cases: system-autonomous firing, object-autonomous firing, and synchronised firing. Again, a transition can only occur

autonomously if there exists no synchronisation partner in ρ, otherwise only synchronous firing is possible.

Definition 3. *Let $OS = (N, \hat{N}, \rho)$ be an unary elementary object system. Let $M \in \mathcal{M}$ be a marking wrt. reference semantics of OS. Firing of transitions can take place in three forms:*

1. *System-autonomous: A transition $t \in T_{\bar{\rho}}$ is activated in $\mathbf{M} = M + \hat{M}$, iff $M \geq pre(t)$. The successor marking is $\mathbf{M'} = M' + \hat{M}$ where $M' = M - pre(t) + post(t)$.*
2. *Synchronous: A pair $(t, \hat{t}) \in \rho$ is activated in $\mathbf{M} = M + \hat{M}$, iff $M \geq pre(t)$ and $\hat{M} \geq \hat{pre}(\hat{t})$. The successor marking is $\mathbf{M'} = M' + \hat{M'}$ where $M' = M - pre(t) + post(t)$ and $\hat{M'} = \hat{M} - \hat{pre}(\hat{t}) + \hat{post}(\hat{t})$.*
3. *Object-autonomous: A transition $\hat{t} \in \hat{T}_{\bar{\rho}}$ is activated in $\mathbf{M} = M + \hat{M}$, iff $\hat{M} \geq \hat{pre}(\hat{t})$. The successor marking is $\mathbf{M'} = M' + \hat{M}$ where $\hat{M'} = \hat{M} - \hat{pre}(\hat{t}) + \hat{post}(\hat{t})$.*

Since there is exactly one system-net and one object-net the whole Ueos can easily be described by the union of system and object-net, where synchronising transitions $(t, \hat{t}) \in \rho$ are fused.

Definition 4. *Let $OS = (N, \hat{N}, \rho)$ be an* Ueos. *The* fusion *$fuse(OS)$ is defined as the P/T-net:*

$$fuse(OS) = \Big((P \cup \hat{P}), (T_{\bar{\rho}} \cup \hat{T}_{\bar{\rho}} \cup \rho), pre^f, post^f, (M_0 + \hat{M}_0)\Big)$$

where

$$pre^f(\tau) = \begin{cases} pre(\tau), & if\ \tau \in T_{\bar{\rho}} \\ \hat{pre}(\tau), & if\ \tau \in \hat{T}_{\bar{\rho}} \\ pre(t) + \hat{pre}(\hat{t}), & if\ \tau \in \rho \wedge \tau = (t, \hat{t}) \end{cases}$$

and

$$post^f(\tau) = \begin{cases} post(\tau), & if\ \tau \in T_{\bar{\rho}} \\ \hat{post}(\tau), & if\ \tau \in \hat{T}_{\bar{\rho}} \\ post(t) + \hat{post}(\hat{t}), & if\ \tau \in \rho \wedge \tau = (t, \hat{t}). \end{cases}$$

It is easy to see, that a transition from $\tau \in \mathbf{T} = (T_{\bar{\rho}} \cup \hat{T}_{\bar{\rho}} \cup \rho)$ is activated for an Ueos OS if it is activated in its fusion $fuse(OS)$. So, the P/T-net $fuse(OS)$ can be used to define the semantics of an Ueos. However, $fuse(OS)$ has in general a much more greater size (counting the number of places and transitions) than sizes of system and object-net together – in the worst case, where all transitions synchronise with all possible partners, i.e. $\rho = T \times \hat{T}$, the fusion $fuse(OS)$ has $|T| \cdot |\hat{T}|$ transitions - compared to $|T| + |\hat{T}|$ for the components in isolation. So, the Ueos is usually easier to understand compared to its fusion.

Proposition 1. *Let OS be an* Ueos. *A a transition $\tau \in \mathbf{T}$ is activated in OS wrt. reference semantics iff it is activated in $fuse(OS)$:*

$$\mathbf{M} \xrightarrow[OS]{\tau} \mathbf{M'} \iff \mathbf{M} \xrightarrow[fuse(OS)]{\tau} \mathbf{M'}$$

Proof. Immediately from Definition 3. □

UEOS *as a Canonical Extension* The definition is a conservative extension of P/T-nets, since an UEOS with an object-net without any transitions behaves like the system-net, i.e. like a P/T-net.

Proposition 2. *Let* $OS_\emptyset = (N_\emptyset, \hat{N}_\emptyset, \rho_\emptyset)$ *be an* UEOS *with empty sets of places and transitions for the object-net:* $\hat{P} = \hat{T} = \emptyset$*. A transition* $\tau \in (T_{\bar{\rho}} \cup \hat{T}_{\bar{\rho}} \cup \rho)$ *is enabled in the object system* $OS_\emptyset$ *iff it is enabled in its system-net* N*.*

$$\mathbf{M} \xrightarrow[OS_\emptyset]{\tau} \mathbf{M}' \iff \mathbf{M} \xrightarrow[N]{\tau} \mathbf{M}'$$

Proof. In $OS_\emptyset$, neither synchronous firing nor object-autonomous firing are possible, because $\hat{T} = \emptyset$. Due to Def. 4 we have $fuse(OS_\emptyset) = N_\emptyset$. Since $MS(\emptyset) = \{0\}$, we only have to deal with empty object-net markings $\hat{M} = 0$ and $\mathbf{M} = M$. Due to Proposition 1 a transition in $OS_\emptyset$ is activated iff it is in $fuse(OS_\emptyset) = N_\emptyset$. □

3.3 Value Semantics

Value semantics considers net-tokens as values. For value semantics markings are described by nested multisets. For UEOS the nesting level is fixed with one level of nesting.

Definition 5. *A* marking *wrt. value semantics of* OS *is a multiset* $\mathbf{M} \in \mathcal{M}_v := MS(P \times MS(\hat{P}))$*. The* initial marking *of a* UEOS OS *wrt. value semantics is* $\mathbf{M}_0 = (p, \hat{M}_0)$ *for* $M_0 = p$*.*

By using projections on the first or last component of a UEOS marking $\mathbf{M}$, it is possible to compare object system markings. The projection $\Pi^1(\mathbf{M})$ on the first component abstracts away the substructure of a net token, while the projection $\Pi^2(\mathbf{M})$ on the second component can be used as the abstract marking of the net tokens without considering their local distribution within the system-net.

Definition 6. *Let* $\mathbf{M} = \sum_{i=1}^{n}(p_i, \hat{M}_i)$ *be a marking of a* UEOS $OS = (N, \hat{N}, \rho)$*.*

1. *Abstraction from the token substructure:*

$$\Pi^1(\sum_{i=1}^{n}(p_i, \hat{M}_i)) = \sum_{i=1}^{n} p_i$$

2. *Summation of the distributed object-net copy's markings:*

$$\Pi^2(\sum_{i=1}^{n}(p_i, \hat{M}_i)) = \sum_{i=1}^{n} \hat{M}_i$$

The UEOS firing rule is defined for three cases: system-autonomous firing, object-autonomous firing, and synchronised firing. A transition can only occur autonomously if there exists no synchronisation partner in ρ, i.e. if $\tau \in (T_{\bar{\rho}} \cup \hat{T}_{\bar{\rho}})$. Otherwise if $\tau \in \rho$, only synchronous firing is possible.

The autonomous firing of a system-net transition t removes net tokens in the pre-conditions together with their individual internal markings. Since the markings of UEOS are higher-order multisets, we have to consider terms $\mathbf{PRE} \in \mathcal{M}_v$ that correspond to the pre-set of t in their first component: $\Pi^1(\mathbf{PRE}) = \mathrm{pre}(t)$. In turn, a multiset $\mathbf{POST} \in \mathcal{M}_v$ is produced, that corresponds with the post-set of t in its first component. Thus, the successor marking is $\mathbf{M'} = \mathbf{M} - \mathbf{PRE} + \mathbf{POST}$, in analogy to the successor marking $M' = M - \mathrm{pre}(t) + \mathrm{post}(t)$ of P/T-nets. The firing of t must also obey the *object marking distribution condition* $\Pi^2(\mathbf{PRE}) = \Pi^2(\mathbf{POST})$, ensuring that the sum of markings in the copies of a net token is preserved.[1]

An object-net transition $\hat{t}$ is enabled autonomously if in $\mathbf{M}$ there is an addend $(p, \hat{M})$ in the sum and $\hat{M}$ enables $\hat{t}$. For synchronous firing, a combination of both is required.

Definition 7. *Let $OS = (N, \hat{N}, \rho)$ be an* UEOS. *Let $M \in \mathcal{M}_v$ be a marking wrt. value semantics of OS. For each transition $\tau \in \mathbf{T}$ we define:*

$$\begin{aligned}\mathbf{M} \xrightarrow[OS]{\tau} \mathbf{M'} \iff \exists \mathbf{PRE}, \mathbf{POST} \in \mathcal{M}_v : \mathbf{M} \geq \mathbf{PRE} \wedge \\ VP_\tau(\mathbf{PRE}, \mathbf{POST}) \wedge \\ \mathbf{M'} = \mathbf{M} - \mathbf{PRE} + \mathbf{POST}\end{aligned}$$

Additionally, the predicate has to be left-total and respects the conditions:

1. *System-autonomous firing, $\tau = t \in T_{\bar{\rho}}$:*

$$\begin{aligned}VP_t(\mathbf{PRE}, \mathbf{POST}) \Longrightarrow \Pi^1(\mathbf{PRE}) = pre(t) \wedge \Pi^1(\mathbf{POST}) = post(t) \wedge \\ \Pi^2(\mathbf{POST}) = \Pi^2(\mathbf{PRE})\end{aligned}$$

2. *Synchronous firing, $\tau = (t, \hat{t}) \in \rho$:*

$$\begin{aligned}VP_{(t,\hat{t})}(\mathbf{PRE}, \mathbf{POST}) \Longrightarrow \Pi^1(\mathbf{PRE}) = pre(t) \wedge \Pi^1(\mathbf{POST}) = post(t) \wedge \\ \Pi^2(\mathbf{PRE}) \geq \hat{pre}(\hat{t}) \wedge \\ \Pi^2(\mathbf{POST}) = \Pi^2(\mathbf{PRE}) - \hat{pre}(\hat{t}) + \hat{post}(\hat{t})\end{aligned}$$

3. *Object-autonomous firing, $\tau = \hat{t} \in \hat{T}_{\bar{\rho}}$:*

$$\begin{aligned}VP_{\hat{t}}(\mathbf{PRE}, \mathbf{POST}) \Longrightarrow \exists p \in P : \Pi^1(\mathbf{PRE}) = \Pi^1(\mathbf{POST}) = p \wedge \\ \Pi^2(\mathbf{PRE}) \geq \hat{pre}(\hat{t}) \wedge \\ \Pi^2(\mathbf{POST}) = \Pi^2(\mathbf{PRE}) - \hat{pre}(\hat{t}) + \hat{post}(\hat{t})\end{aligned}$$

Note, that the three cases can be simplified to one single case if we consider "virtual" transitions: For system-atonomes firing we use the virtual transitions $\hat{\epsilon}$ of the object net with $\hat{\mathrm{pre}}(\hat{\epsilon}) = \hat{\mathrm{post}}(\hat{\epsilon}) = 0$. For object-atonomes firing we use the family of transiitons $\epsilon_P := \{\epsilon_p \mid p \in P\}$ with $\mathrm{pre}(\epsilon_p) = \mathrm{post}(\epsilon_p) = p$. It is

[1] This represents the main difference of UEOS compared with Valk's object systems [Val98], which require that each net token in **POST** hold all tokens $\Pi^2(\mathbf{PRE})$ of the net tokens from the pre-set. This kind of multiplication of system resources is inhibited in UEOS. Note, that almost all propositions exploit this symmetry.

easy to observe, that using these virtual events the autonomous cases can be deduced from the synchronous case.

In contrast to P/T-nets, the successor marking is not uniquely defined for Ueos, due to their vertical substructure. The firing rule can be made deterministic by using the pair $(\mathbf{PRE}, \mathbf{POST})$ as an index for transitions in the system-net, i.e. considering the infinite family $t_{(\mathbf{PRE},\mathbf{POST})}$ instead of t.

The following properties characterise the symmetries of Ueos.

Proposition 3. *Let OS be an* Ueos *as in Def. 1.*

1. *The abstract behaviour – determined by the projection Π^1 – on the system level of a* Ueos *corresponds to the behaviour of the system-net viewed as a P/T-net:*
   $\mathbf{M} \xrightarrow[OS]{t} \mathbf{M}'$ *implies* $\Pi^1(\mathbf{M}) \xrightarrow[N]{t} \Pi^1(\mathbf{M}')$*, and* $\Pi^1(\mathbf{M}')$ *is uniquely determined.*
2. *The distributed object-net marking is invariant under autonomous actions of the system-net:*
   $\mathbf{M} \xrightarrow[OS]{t} \mathbf{M}'$ *implies* $\Pi^2(\mathbf{M}) = \Pi^2(\mathbf{M}')$.
3. *The behaviour of the distributed object-net is determined by Π^2 and is a sub-behaviour of the object-net viewed as a P/T-net:*
   $\mathbf{M} \xrightarrow[OS]{\hat{t}} \mathbf{M}'$ *implies* $\Pi^2(\mathbf{M}) \xrightarrow[\hat{N}]{\hat{t}} \Pi^2(\mathbf{M}')$*, and* $\Pi^2(\mathbf{M}')$ *is uniquely determined.*
4. *The abstract state of the system-net is invariant under object-autonomous actions:*
   $\mathbf{M} \xrightarrow[OS]{\hat{t}} \mathbf{M}'$ *implies* $\Pi^1(\mathbf{M}) = \Pi^1(\mathbf{M}')$.
5. *Synchronisation is an action composed of two sub-actions:*
   $\mathbf{M} \xrightarrow[OS]{(t,\hat{t})} \mathbf{M}'$ *implies* $\Pi^1(\mathbf{M}) \xrightarrow[N]{t} \Pi^1(\mathbf{M}')$ *as well as* $\Pi^2(\mathbf{M}) \xrightarrow[\hat{N}]{\hat{t}} \Pi^2(\mathbf{M}')$. *Furthermore,* $\Pi^1(\mathbf{M}')$ *and* $\Pi^2(\mathbf{M}')$ *are uniquely determined.*

Proof. Immediate from Definition 7. □

Note, that not all properties hold in he semantics presented in [Val98], Since each net token in **POST** hold all tokens $\Pi^2(\mathbf{PRE})$ of the net tokens from the pre-set (i.e. tokens are duplicated) the distributed object-net marking $\Pi^2(\mathbf{M})$ cannot be preserved. So, Prop. 3 (2) and (5) become invalid for the semantics presented in [Val98].

Ueos *as a Canonical Extension* By using a degenerated Ueos, which has neither places nor transitions, it can easily be shown that Ueos are a canonical extension of P/T-nets with respect to interleaving semantics (i.e. firing sequences). We thereby define an Ueos whose tokens have no inner structure. Therefore, this Ueos is equivalent to a P/T-net.

Proposition 4. *Let $OS_\emptyset$ be an* Ueos *with empty sets of places and transitions for the object-net: $\hat{P} = \hat{T} = \emptyset$. A transition t is enabled in the object system*

$OS_\emptyset$ iff it is enabled in its system-net N.

$$\mathbf{M} \xrightarrow[OS_\emptyset]{t} \mathbf{M}' \iff \mathbf{M} \xrightarrow[N]{t} \mathbf{M}'$$

Proof. In $OS_\emptyset$, neither synchronous firing nor object-autonomous firing are possible, because $\hat{T} = \emptyset$. Since $MS(\emptyset) = \{0\}$, we only have to deal with empty object-net markings $\hat{M} = 0$, the object marking distribution condition is always met. Since VP is left-total, there exists a successor marking $\mathbf{M}'$ for $\mathbf{M}$. The equivalence follows from Proposition 3 (1), stating that the projection $\Pi^1(\mathbf{M})$ is isomorphic to the marking M of the system-net N. □

3.4 Relationship of Value and Reference Semantics

The translation of the notion of "activation" from value to reference semantics are based on the translation from value-markings to reference-markings. Nested multisets can be mapped directly to unnested ones. Define $\phi : \mathcal{M}_v \to \mathcal{M}_r$ by the linear extension of $\phi((p, \hat{M})) = p + \hat{M}$.

Proposition 5. *Let OS be an* Ueos *as in Def. 1 and let $\mathbf{M}, \mathbf{M}' \in \mathcal{M}_v$ be markings wrt. value semantics. If $\mathbf{M} \xrightarrow[OS]{w} \mathbf{M}'$, $w \in \mathbf{T}$ is a possible firing wrt. value semantics then $\phi(\mathbf{M}) \xrightarrow[OS]{w} \phi(\mathbf{M}')$ is a possible firing sequence wrt. reference semantics.*

Proof. From the definitions of the firing rules in Def. 7 (value semantics) and Def. 3 (reference semantics) it can be seen directly, that activation wrt. value semantics implies activation wrt. reference semantics. □

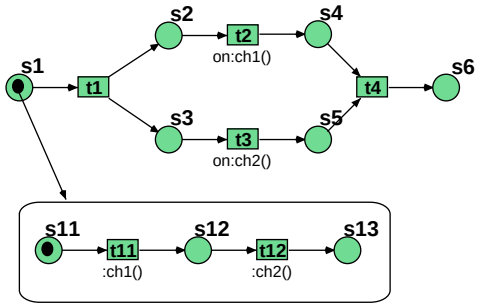

Fig. 4. An Example Ueos

The converse direction does not hold, which can be demonstrated using the Ueos in Fig. 4. For reference semantics the place s_1 initially contains a reference to the object-net: $\mathbf{M} = s_1 + \hat{s}_{11}$. Firing of t_1 duplicates this reference onto s_2 and s_3 resulting in the marking $\mathbf{M}_1 = s_2 + s_3 + \hat{s}_{11}$. This marking activates the transition pair $(t_2, \hat{t}_{11})$ while $(t_3, \hat{t}_{12})$ is not. The resulting marking

is $\mathbf{M}_2 = s_4 + s_3 + \hat{s}_{12}$. Since the effect in the object-net is visible in the whole system, the pair $(t_3, \hat{t}_{12})$ is now activated. Firing leads to $\mathbf{M}_3 = s_4 + s_5 + \hat{s}_{13}$.

For value semantics we have $\mathbf{M} = (s_1, \hat{s}_{11})$. Here $\phi(\mathbf{M}) = s_1 + \hat{s}_{11}$ is the corresponding marking wrt. reference semantics. Firing of t_1 can result in in the marking $\mathbf{M}_1 = (s_2, \hat{s}_{11}) + (s_3, 0)$ – corresponding to $\phi(\mathbf{M}_1) = s_2 + s_3 + \hat{s}_{11}$ for reference semantics. This marking activates the transition pair $(t_2, \hat{t}_{11})$ while $(t_3, \hat{t}_{12})$ is not. The resulting marking is $\mathbf{M}_2 = (s_4, \hat{s}_{12}) + (s_3, 0)$. Since the effect in the object net is only local the pair $(t_3, \hat{t}_{12})$ is not activated. So $w = t_1(t_2, \hat{t}_{11})(t_3, \hat{t}_{12}) \in \mathbf{T}^*$ is a possible firing sequence for reference but not for value semantics.

The difference of reference and value semantics is the concept of "location" for net tokens which is explicit for value but not for reference semantics, since it is unclear which reference can be considered as the location of a net token. The localisation of net tokens is expressed by a mapping from reference semantics markings to value semantics ones.

Definition 8. *A mapping $\sigma : \mathcal{M}_r \to \mathcal{M}_v$ is called a* localisation *iff for each $\mathbf{M} \in \mathcal{M}_r$ with $\mathbf{M} = M_1 + \hat{M}_2$, $M_1 \in MS(P)$, $\hat{M}_2 \in MS(\hat{P})$ the following holds:*

$$\Pi^1(\sigma(\mathbf{M}_r)) = M_1 \qquad \text{and} \qquad \Pi^2(\sigma(\mathbf{M}_r)) = \hat{M}_2$$

A localisation σ is compatible *wrt. the firing rule iff for each $\mathbf{M}, \mathbf{M}' \in \mathcal{M}_r$ and $w \in \mathbf{T}$ we have:*

$$\mathbf{M}_r \xrightarrow[OS]{w} \mathbf{M}'_r \Longrightarrow \sigma(\mathbf{M}_r) \xrightarrow[OS]{w} \sigma(\mathbf{M}'_r)$$

For the special case that the system-net is a state machine the ambiguity of locations cannot appear, since there always is exactly one reference.

Definition 9. *An* Ueos *OS is called* simple *iff the system-net N is a state machine, i.e. $|M_0| = 1$ and for all $t \in T$ we have $|pre(t)| \leq 1$ and $|post(t)| \leq 1$.*

For a simple Ueos there is exactly one localisation mapping $\sigma : \mathcal{M}_r \to \mathcal{M}_v$: Since N is a state machine for each $\mathbf{M} \in \mathcal{M}_r$ with $\mathbf{M} = M + \hat{M}$ we have exactly one net token reference: $|M| = 1$. So, with $M = p$ the only map with $\Pi^1(\sigma(\mathbf{M}_r) = \hat{M}$ and $\Pi^2(\sigma(\mathbf{M}_r) = \hat{M}$ is $\sigma_{sm}(p + \hat{M}) := (p, \hat{M})$. For a simple Ueos reference and value semantics coincide.

Proposition 6. *Let OS be a simple* Ueos*. Then σ_{sm} is a localisation, that is compatible with the firing rule, i.e. $\mathbf{M}_r \xrightarrow[OS]{w} \mathbf{M}'_r \Longrightarrow \sigma_{sm}(\mathbf{M}_r) \xrightarrow[OS]{w} \sigma_{sm}(\mathbf{M}'_r)$ for all $w \in \mathbf{T}$.*

Proof. Follows from the property of a localisation map σ (Def. 8) and the definition of the value semantics firing rule (Def. 7) and of the reference semantics firing rule (Def. 3). □

The general condition for a firing sequence wrt. reference semantics being also possible for value semantics is given in [KF03] where the concept of locality that is present for value semantics but not for reference semantics is combined with Petri net processes.

3.5 Invariants in Ueos

The compositionality of invariants is interesting for Ueos. In the following we investigate how the invariant calculus of P/T-nets extends for Ueos. Let $N = (P, T, \mathrm{pre}, \mathrm{post}, M_0)$ be a P/T-net. The incidence matrix Δ is defined by $\Delta(p,t) := \mathrm{post}(t)(p) - pre(t)(p)$. A P-invariant $\boldsymbol{i} \in \mathbb{Z}^{|P|}, \boldsymbol{i} \neq \mathbf{0}$ is a vector that fulfils $\boldsymbol{i} \cdot \Delta = \mathbf{0}$. Then every reachable marking $\boldsymbol{M}$ fulfils the linear equation $\boldsymbol{i} \cdot \boldsymbol{M} = \boldsymbol{i} \cdot \boldsymbol{M}_0$.

The following theorem states that invariants for a Ueos can easily be composed from the invariants of the components.

Proposition 7. *Let $OS = (N, \hat{N}, \rho)$ be an Ueos as in Def. 1 and let $\boldsymbol{i} \in \mathbb{Z}^{|P|}$ be an invariant of the system-net and $\hat{\boldsymbol{i}} \in \mathbb{Z}^{|\hat{P}|}$ one of the object-net.*

Let $\mathbf{M} \in \mathcal{M}_r$ be a reference semantics marking. Then $\boldsymbol{i} \cdot \mathbf{M}_{|P} = \boldsymbol{i} \cdot \mathbf{M}_{0|P}$ and $\hat{\boldsymbol{i}} \cdot \mathbf{M}_{|\hat{P}} = \hat{\boldsymbol{i}} \cdot \mathbf{M}_{0|\hat{P}}$ holds for every reachable marking $\mathbf{M} \in \mathcal{M}_r$ wrt. reference semantics. Additionally, the vector $(\boldsymbol{i}, \hat{\boldsymbol{i}}) \in \mathbb{Z}^{|P|+|\hat{P}|}$ is an invariant of the fusion net fuse(OS).

Let $\mathbf{M} \in \mathcal{M}_v$ be a value semantics marking. Then $\boldsymbol{i} \cdot \Pi^1(\mathbf{M}) = \boldsymbol{i} \cdot \Pi^1(\mathbf{M}_0)$ and $\hat{\boldsymbol{i}} \cdot \Pi^2(\mathbf{M}) = \hat{\boldsymbol{i}} \cdot \Pi^2(\mathbf{M}_0)$ holds for every reachable marking $\mathbf{M} \in \mathcal{M}_v$ wrt. value semantics.

Proof. For reference semantics the propositions follows from Proposition 1.

For value semantics: For a system-autonomous step $\mathbf{M} \xrightarrow{\tau} \mathbf{M}'$ we have $\Pi^1(\mathbf{M}) \xrightarrow[N]{\tau} \Pi^1(\mathbf{M}')$ by Prop. 3(1). Since $\boldsymbol{i}$ is an invariant of N we have $\boldsymbol{i} \cdot \Pi^1(\mathbf{M}) = \boldsymbol{i} \cdot \Pi^1(\mathbf{M}')$. Also, we have $\hat{\boldsymbol{i}} \cdot \Pi^2(\mathbf{M}) = \hat{\boldsymbol{i}} \cdot \Pi^2(\mathbf{M}_0)$, since $\Pi^2(\mathbf{M}) = \Pi^2(\mathbf{M}')$ by Prop. 3(2). Analogously with Prop. 3(3) and (4) for object-autonomous and steps and with Prop. 3(5) for synchronisations. □

Before proving further results for Ueos wrt. decidability issues in Section 5 related models are introduced.

4 Extended Object-Net Formalisms

In this section Object-Net formalisms that extend Ueos are investigated. These are the Minimal Object-Oriented Nets, Nested Petri Nets, and Anonymous Object-Net Systems. Also Mobility Calculi are studied due to their close connection to nets within nets.

4.1 Minimal Object-Oriented Nets

Minimal Object-Oriented Nets (minimal OO-nets for short) [Kum00] are defined to act as a general abstraction for object-oriented extensions of Petri nets, i.e. Petri nets that deal with the objects of a object-oriented programming language (cf. [ACR00]). Minimal OO-nets makes no assumptions over the inner structure of objects – the only requirement is, that objects can be created at run-time and objects have a unique identity that can be checked for

equality. These requirements should be fulfilled by every object-oriented Petri net formalism.

Note, that also "nets within nets" can be seen as an object-oriented formalism if we provide the net tokens with identities and regard them as objects.

Definition 10. *A* minimal OO-net *is a tuple $N = (P, T, F, X, w, e)$, where P and T are disjoint sets, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, X is a set of variables, w is a variable assign functions on F into X and $e \in X$ is a special variable. A marking m is a function on P into multisets over arbitrary objects.*

The arcs of a minimal OO-net carry variables, that are bound to objects when firing a transition. The special variable $e \in X$ is used in a way, that – when firing a transition – it is bound to a identity, that is not present in the current marking, i.e. a new object is created.

Definition 11. *Let m be a marking and t a transition. Let f a variable binding function defined for all $x \in X$. The transition t is* activated *iff $\forall p \in P : (p,t) \in F \Longrightarrow f(w(p,t)) \in m(p)$ ("enough objects") and $\forall p \in P : f(e) \notin m(p)$ ("e is unused").*

The successor marking *m'' is defined the following way: Let $m'(p) = m(p) - f(w(p,t))$ if $(p,t) \in F$ and $m'(p) = m(p)$ otherwise and let $m''(p) = m'(p) + f(w(t,p))$ if $(t,p) \in F$ and $m''(p) = m'(p)$ otherwise.*

Reference Nets [Kum02] are a net formalism that is implemented in the Renew-tool [KWD03]. Renew deals with two kinds of objects: objects can be instances of Java classes – the inscription language of Reference Nets – and instances of Reference Nets. Reference nets uses reference semantics (therefore the name) for the net tokens. Net tokens have a unique identity and new net tokens can be created at run-time (whereas Ueos deals with exactly only one identity).

4.2 Anonymous Object-Net Systems

In the following a generalised model of object-net systems is defined, which drops the restriction to exactly two levels of nesting: Anonymous[2] Object-Net Systems (Ons) are defined to give a precise definition of nets within nets using algebraic specifications. As [SMÖ01] mentioned in their outlook it is quite natural extension of algebraic Petri nets [Rei91] to allow tokens to be *active* which is impossible for algebraic Petri nets. The canonic way for this extension is to consider nets as these active tokens. In the following the algebraic description of object-net systems is given in rewriting logic [Mes92] (see the Appendix) using the syntax of Maude [CDE+99]. Let $\mathcal{N}$ be a finite set of net sorts in the following.[3] The black token net N_* is defined as the object-net with no places and no transitions: $P(N_*) = T(N_*) = \emptyset$.

[2] These object-net systems are called "anonymous" since the net tokens carry no identity. Identities are omitted, since we are interested in expressiveness bounds and every Petri net formalism using an unbounded set of identities is Turing powerful (see Prop. 12 below).

[3] Therefore the notation conventions for elements in Ueos (like $\mathbf{x}$, $\hat{x}$, and x) are not applicable anymore.

Channels Object-nets synchronise via a set of channels $V = \bigcup_{N\in\mathcal{N}} V_N$ with disjoint V_N. Channels are either directed downwards (called *downlinks* – the "synchronising" side, where the net-token is known) or upwards (*uplinks* – the "synchronised" side).[4] The labelling function $\lambda^{\uparrow} : T \to MS(V)$ maps each transition to a multiset of uplinks while $\lambda^{\downarrow} : T \to MS(V)$ maps a transition to a multiset of downlinks. Using multisets of labels allows to describe multiple synchronisation. The neutral element 0 means that no synchronisation is needed. There is at most one uplink, i.e. $|\lambda^{\uparrow}(t)| \leq 1$ for each transition $t \in T$, which is required to obtain a tree-like synchronisation structure.

Nets as Tokens Markings are multisets of net-tokens (following the "Petri nets are monoids" paradigm [MM90]). Define the sorts of tokens and markings with the usual multiset addition (being associative and commutative) and the neutral element 0_N:

`sorts` $Token_N$ MS_N .
`subsort` $Token_N < MS_N$.
`op` $0_N : \to MS_N$.
`op _ + _ :` $MS_N \times MS_N \to MS_N$ `[assoc comm id:` 0_N`]` .

Object-nets describe the structure of net-tokens. Net-tokens are obtained if a marking is considered as one single entity. This is done by the operator $\mathrm{net}_N : MS_N \to N$.

Tokens are described by their value and the place that is marked. It is natural to use the place name as operators. Note that these operators are needed to formulate the non-zero multisets. Each place $p \in P$ is mapped by $d : P \to \mathcal{N}$ to a place sort. Define for all $p \in P(N)$ the place operators: $p : d(p) \to Token_N$. In the following the token term $p(\mathrm{net}_{d(p)}(M))$ is abbreviated as $p[M]$. Let "$\bullet$" be the abbreviation of [0] for N_*. We assume, that the initial marking of all nets contains only black-tokens: $M_0(N)(p) > 0 \Longrightarrow d(p) = N_*$ for all $N \in \mathcal{N}$. The *initialisation* of a net-token given the net sort N is defined as:

$$init_N := \mathrm{net}_N\big(\sum\nolimits_{p\in P(N)} M_0(N)(p) \cdot p(\bullet)\big)$$

Define for all $v \in V_N$ the constants $v^{?!}, v^{!?} : \to Token_N$. Informally the place $v^{?!}$ stands for the start of the atomically executed sub-synchronisation and $v^{!?}$ stands for its end.

Object-Net Systems An anonymous Object-Net System (Ons) consists of a set of Petri Nets. For simplicity we denote this set also by $\mathcal{N}$, so N denotes a sort as well as a net. This re-use is harmless, since it can be resolved by introducing a set $\{ON_N : N \in \mathcal{N}\}$ isomorphic to $\mathcal{N}$. One net $N_{sn} \in \mathcal{N}$ is the system-net, building the top level of the system. It is assumed that places and transitions are disjoint for all nets and all nets use a common set of up- and downlinks V.

Definition 12. *An* anonymous Object-Net System *is the tuple*

$$ONS = (\mathcal{N}, d, V, \lambda)$$

[4] Note, that a horizontal synchronisation of two net-tokens occupying the same place can be simulated by attaching a transition to that place which synchronises twice vertically. Therefore synchronisation is restricted to the vertical case.

1. *$\mathcal{N} = \{N_1, \dots N_n\}$ is a set of pairwise disjoint P/T-nets. $\mathcal{N}$ includes the black token net $N_* \in \mathcal{N}$ and the system-net $N_{sn} \in \mathcal{N}$. Let P be the union of all components: $P := \bigcup_{N \in \mathcal{N}} P(N)$. Analogously for T, pre, and post.*
2. *$d : P \to \mathcal{N}$ is the place typing.*
3. *$V = \bigcup_{N \in \mathcal{N}} V_N$ is the set of channels.*
4. *$\lambda = (\lambda^{\uparrow} : T \to MS(V), \lambda^{\downarrow} : T \to MS(V))$ are mappings of transitions to multisets of up- resp. downlinks. There is at most one uplink, i.e. $|\lambda^{\uparrow}(t)| \leq 1$ for all transition $t \in T$.*

   The initial marking is $M_0 = init_{N_{sn}}$.

An Object Petri Net is called *pairwise synchronised* iff every transition t has either uplinks or downlinks but not both (similar to process calculi):

$$\forall t \in T : (|\lambda^{\uparrow}(t)| > 0 \Longrightarrow |\lambda^{\downarrow}(t)| = 0) \wedge (|\lambda^{\downarrow}(t)| > 0 \Longrightarrow |\lambda^{\uparrow}(t)| = 0)$$

This condition is needed to obtain a finite set of possible synchronisations.

Firing Rule The presentation of Petri nets as tokens leads to the problem, how synchronised actions spanning over several net levels should be formalised. Due to the tree-like structure of synchronisation there are infinitely many possible synchronisations in general, so we cannot formalise each synchronisation as a transition. Instead we formalise sub-synchronisations as conditional rewrites. Define for all $t \in T$ the following conditional rule:

$$\begin{aligned} & t : \lambda^{\uparrow}(t)^{?!} + \sum_{p \in {}^\bullet t} \sum_{i=1}^{W(p,t)} p[A_{p,t,i} + B_{p,t,i}] \\ & \to \lambda^{\uparrow}(t)^{!?} + \sum_{p \in t^\bullet} \sum_{j=1}^{W(t,p)} p[A'_{p,t,j} + B'_{p,t,j}] \quad \text{if } \psi_1(t) \wedge \psi_2(t) \wedge \psi_3(t) \end{aligned}$$

1. The uplink (if any) $v \in V$ is denoted as $\lambda^{\uparrow}(t)^{?!}$ in the pre-set and as $\lambda^{\uparrow}(t)^{!?}$ in the post-set.
2. The markings of net-tokens are distributed into two groups of multisets: $A_{p,t,i}, A'_{p,t,j}$ for the tokens that are transported and $B_{p,t,i}, B'_{p,t,j}$ for the tokens that are used for synchronisation. Unprimed variables are used for variables of the pre-set and single primed ones for in the post-set.
   - For all $p \in {}^\bullet t$ and $i \in \{1, \dots, W(p,t)\}$: $A_{p,t,i}, B_{p,t,i} : MS_{d(p)}$.
   - For all $p \in t^\bullet$ and $j \in \{1, \dots, W(t,p)\}$: $A'_{p,t,j}, B'_{p,t,j} : MS_{d(p)}$.
   - For all $v \in V_N$ and $l \in \{1, \dots \lambda^{\downarrow}(t)(v)\}$: $M_{v,l}, M'_{v,l} : MS_N$.
3. The markings of the net-tokens are related by the requirement, that firing preserves the overall sum of all transported net-tokens. The sum is preserved for each net sort N:

$$\psi_1(t) := \forall N \in \mathcal{N} : \sum_{p \in ({}^\bullet t \cap d^{-1}(N))} \sum_{i=1}^{W(p,t)} A_{p,t,i} = \sum_{p \in (t^\bullet \cap d^{-1}(N))} \sum_{j=1}^{W(t,p)} A'_{p,t,j}$$

4. The sub-synchronisations fire $M_{v,l}$ to $M'_{v,l}$. These markings are distributed on the synchronisation markings $B_{p,t,i}$ for $M_{v,l}$. Analogously, the $B'_{p,t,j}$ must sum up to $M'_{v,l}$:

$$\psi_2(t) := \forall N \in \mathcal{N} : \sum_{v \in V_N} \sum_{l=1}^{\lambda^{\downarrow}(t)(v)} M_{v,l} = \sum_{p \in ({}^{\bullet}t \cap d^{-1}(N))} \sum_{i=1}^{W(p,t)} B_{p,t,i} \wedge \sum_{v \in V_N} \sum_{l=1}^{\lambda^{\downarrow}(t)(v)} M'_{v,l} = \sum_{p \in (t^{\bullet} \cap d^{-1}(N))} \sum_{j=1}^{W(t,p)} B'_{p,t,j}$$

5. Each downlink $v \in V$ with $\lambda^{\downarrow}(t)(v) > 0$ creates one sub-synchronisation on this channel in the conditional:

   $$\psi_3(t) := \forall N \in \mathcal{N} : \forall v \in V_N : \forall l \in \{1, \ldots, \lambda^{\downarrow}(t)(v)\} : v^{?!} + M_{v,l} \rightarrow v^{!?} + M'_{v,l}$$

   By this construction every outgoing channel results in a sub-synchronisation which can be executed by the corresponding incoming channel side.

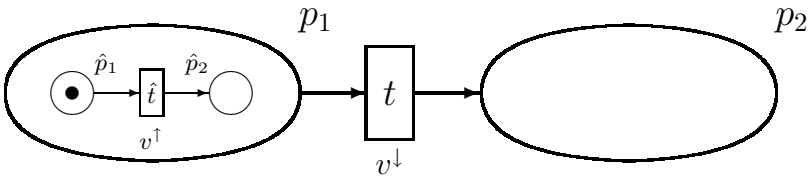

Fig. 5. An Object-Net System

Example 1. Have a look at a simple example, pictured in Fig. 5: The transition t fires synchronously via the channel v with the object-net transition $\hat{t}$. Synchronous firing of t and $\hat{t}$ wrt. channel v, can be formalised as:

$$t /\!\!/_v \hat{t} : p[m'' + \hat{p}(\bullet)] \rightarrow p'[m'' + \hat{p}'(\bullet)]$$

Handling the inner actions of active tokens in such an explicit way is undesirable of several reasons. For the general case, where the synchronous actions takes place at spanning over a nesting-hierarchy, each possible nesting structure has to be described by one single rule. Instead conditional rewriting rules can be used. The vertically called channel v is used to denote the synchronisation partner. Informally the channel $v^{?!}$ stands for the start of the atomically executed sub-synchronisation and $v^{!?}$ stands for its end:

$$p[m] \rightarrow p'[m'] \text{ if } v^{?!} + m \rightarrow v^{!?} + m'$$

The transition in the object-net is combined with the called channel:

$$\hat{t} : v^{?!} + \hat{p}(\bullet) \rightarrow v^{!?} + \hat{p}'(\bullet)$$

For the binding $m = m'' + \hat{p}(\bullet)$ and $m' = m'' + \hat{p}'(\bullet)$ the conditional is fulfilled.

4.3 Mobility Calculi

For a comparison of our approach with common mobility calculi, we take the Ambient Calculus [CGG99] as an example. In the Ambient Calculus a process term P with locality n is denoted as $n[P]$, which corresponds roughly to a net token in the place n which is in a state corresponding to P. Each location N can be left by the action "out n" or entered by "in n".

Let *Names* be a set of names and let $n \in Names$ then the expressions of the ambient calculus are constructed the following way:

$$P ::= \alpha.P \mid (P_1|P_2) \mid \mathsf{new}\ n : P \mid !P \mid n[P]$$

where α denotes leaving (out n), entering (in n), or the destruction (open n) of a locality context:

$$\alpha ::= \text{out } n \mid \text{in } n \mid \text{open } n$$

The semantics is defined by a transition system with the set of expressions as nodes and transitions constructed by the following rules:

in

$$n[\text{in } m.P|Q] \ |m[R] \rightarrow m[n[P|Q] \ |R]$$

out

$$m[n[\text{out } m.P|Q] \ |R] \rightarrow n[P|Q] \ |m[R]$$

open

$$\text{open } m.P \ |m[Q] \rightarrow P|Q$$

Res

$$\frac{P \rightarrow P'}{\mathsf{new}\ x : P \rightarrow \mathsf{new}\ x : P'}$$

Amb

$$\frac{P \rightarrow P'}{n[P] \rightarrow n[P']}$$

Par

$$\frac{P \rightarrow P'}{P \ |Q \rightarrow P' \ |Q}$$

Struct

$$\frac{P \rightarrow P'}{Q \rightarrow Q'} \text{ if } P = Q, P' = Q'$$

where = denotes equality wrt. associativity and/or commutativity of the operators.

$$
\begin{aligned}
P + Q &= Q + P \\
P + (Q + R) &= (P + Q) + R \\
P + P &= P \\
P + 0 &= P \\
P|Q &= Q|P \\
P|(Q|R) &= (P|Q)|R \\
P|0 &= P \\
P.(Q.R) &= (P.Q).R \\
\mathsf{new}\ x : P &= \mathsf{new}\ y : P[x \leftarrow y] \qquad (\alpha\text{-Konversion}) \\
\mathsf{new}\ x : (P|Q) &= P|\mathsf{new}\ x : Q \text{ if } x \notin \mathit{freenames}(P) \\
\mathsf{new}\ x : \mathsf{new}\ y : P &= \mathsf{new}\ y : \mathsf{new}\ x : P \\
!P &= P|!P
\end{aligned}
$$

For a system-net being a state machine (i.e. each transition has at most one input and one output place) the firing of the system-net corresponds to a rewrite in the Ambient Calculus. Assume two places p_1 and p_2 and one object-net token ON. The token ON is moved from place p_1 to p_2 the following way:

$$
\begin{aligned}
& p_1[ON[\text{out } p_1.\text{in } p_2.P|Q]] \mid p_2[] \\
\rightarrow\ & p_1[] \mid ON[\text{in } p_2.P|Q] \mid p_2[] \\
\rightarrow\ & p_1[] \mid p_2[ON[P|Q]]
\end{aligned}
$$

While the relationship of the Ambient Calculus to "nets within nets" seems obvious for system-nets being a state machine, the translation of the general case is not. So nets within nets remain an interesting formalism, since it allows to study concurrency – expressed by independent transitions – and mobility – expressed by the net tokens.

4.4 Nested Petri Nets

Nested Petri Nets (NPN) [Lom00] describe coloured Petri nets where it is allowed to denote nets as arc inscriptions. For Nested Petri Nets it is forbidden to denote the same variable twice in the pre-set, i.e. it is forbidden to combine net tokens. Copying is allowed. This restriction is made to ensure some decidability results, e.g. decidability of termination (cf. the note on Prop. 13 below).

Here, Nested Petri Nets are not investigated in detail because they allow at least for anonymous Ons, so every results for Ons immediately carries over for Nested Petri Nets.

Contrary to other object-net formalism Nested Petri nets denote explicitly the inner marking of net tokens. Due to this it is assumed that that input arcs of transitions carry no constants and no variable occurs twice in the whole of all input expressions. This restriction is needed to obtain a monotonicity property for the firing rule (see below).

5 Decidability Issues

In the following some decidability question related to Petri nets are studied for "nets within nets" formalisms, especially the reachability and the boundedness problem is considered.

5.1 Reachability

It is a well known fact that Petri nets (including EN systems or P/T-nets) are not Turing powerful, since the reachability problem is decidable [May81], whereas the halting problem for Turing machines is not (cf. [HU79]).[5]

Reachability is decidable for 1-safe unary elementary object-nets, since the set of possible markings $\mathcal{M}_v = MS(P \times MS(\hat{P}))$ reduces to $\mathcal{M}_v = 2^{P \times 2^{\hat{P}}}$, which is a finite set. So, $\mathcal{M}_v$ can be used as the set of places, i.e. each place has the form $(p, \hat{M})$. It can be seen directly, that the resulting Petri net has isomorphic marking and firing sequences. The result can be generalised directly for Ueos where system and object-net are n-safe.

For the unbounded case the reachability problem for Ueos wrt. reference semantics is decidable.

Proposition 8. *Reachability is decidable for* Ueos *wrt. reference semantics.*

Proof. The fusion of an Ueos OS results in the P/T-net $fuse(OS)$ which has the same markings and firing sequences (cf. Prop. 1). So, the reachability problem for Ueos reduces to that for P/T-nets. □

On the other hand the reachability problem for Ueos wrt. value semantics is undecidable, since they can simulate nets with transfer arcs. Transfer arcs (cf. [Cia94]) are used to transfer all tokens of one place to another place. So, transfer arcs are related to reset arcs which remove all tokens from one place.

Proposition 9. *Reachability is undecidable for* Ueos *wrt. value semantics.*

Proof. This can be seen by giving a simulation of Petri nets with transfer arcs using Ueos and the fact that the reachability problem for Petri nets with at least two transfer arcs is undecidable (cf. Theorem 11 in [DFS98]). In the simulation the system-net describes the original Petri net N and the object-net acts as a container for the tokens. Figure 6 illustrates the construction: Fig. 6 (a) shows a net with transfer arcs connected with at the transition t_2; Fig. 6 (b) shows the simulating object-net system. The object-net system initially created two copies of the net token an creates the initial marking via the channels init_s1() and init_s2(). Whenever a token is added to the place s_1 by t_1 the system-net removes the object-net on s_1, adds one token inside using the synchronisation mechanism, and puts the object-net back. Similarly for removing tokens from s_2 by t_3. Whenever the whole marking of the place s_1 is transferred to s_2 by t_2 the object net tokens are removed from s_1 and s_2 and the combined marking (i.e. the multiset sum) is places on s_2. □

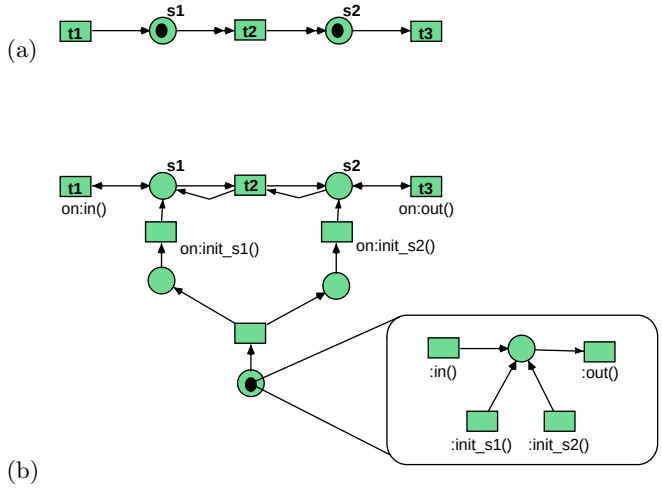

Fig. 6. Simulation of transfer-arcs

As a corrolary we obtain that reachability is also undecidable for Nested Petri Nets, since Nested Petri Nets are a generalisation of Ueos. The direct proof is given in [LS00] where it is shown that Nested Petri Nets can simulate Reset Nets. It is a fact that the reachability problem for Reset Petri Nets with at least two reset arcs is undecidable [DFS98, Theorem 11]. Since reset arcs are more powerful than transfer arcs wrt. decidability of boundedness this result shows that Nested Petri Nets are at least as expressive as Ueos. It is shown in the next section that Nested Petri Nets are more expressive than Ueos.

A related result holds for Linear Logic Petri Nets [Far99], which are Petri nets with Linear Logic formulas as tokens. It is assumed that the token-formulas can be rewritten using the calculus of Linear Logic independently of the firing of transitions. Using undecidability results for deduction in Linear Logic Farwer shows that the reachability problem is *undecidable* for Linear Logic Petri Nets (cf. [Far99]).

5.2 Boundedness

A natural question is whether Ueos are as expressive as Nested Petri Nets. The answer to this question is "no" since it can be shown that boundedness is a decidable property of Ueos, while boundedness is undecidable for Nested Petri Nets.

Proposition 10. *Boundedness is undecidable for Nested Petri Nets.*

Proof. Nested Petri Nets can simulate Reset Nets [LS00] and boundedness for Reset Nets is undecidable [DFS98, Theorem 8]. □

[5] Nevertheless the reachability problem requires exponential space [Lip76]. For a survey on the decidability problem cf. [AK77], for complexity issues cf. [EN94].

In general, the boundedness problem can be decided using the coverability graph construction of Karp and Miller (cf. [KM69]). To construct the coverability graph we are looking for a marking sequence $m_0 \xrightarrow{*} m \xrightarrow{+} m'$ where $m < m'$ holds, i.e. m' covers m on a non-empty subset of places $A \subseteq P$: $m(p) < m'(p)$ for all $p \in A$ and $m(p) = m'(p)$ for all $p \in P \setminus A$. Since Petri nets enjoy the property of monotonicity (i.e. if $m_1 < m_2$ and $m_1 \xrightarrow{*} m_1'$ then there exists a sequence $m_2 \xrightarrow{*} m_2'$ with $m_1' < m_2'$) this sequence can be repeated infinitely often showing that any bound for a place $p \in A$ can be covered, i.e. the place p is unbounded

To apply this technique for Ueos the partial order $\leq$ on multisets can be extended to a partial order on nested multisets.

Definition 13. *Let $\mathbf{M}, \mathbf{M}' \in \mathcal{M}_v$ two nested multisets. Define $\mathbf{M} \preceq \mathbf{M}'$ iff there exists a total and injective mapping f from $\mathbf{M} = \sum_{i=1}^{n}(p_i, \hat{M}_i)$ to $\mathbf{M}' = \sum_{j=1}^{n}(p_j', \hat{M}_j')$ with $f((p_i, \hat{M}_i)) = (p_j', \hat{M}_j')$ implying $p_i = p_j'$ and $\hat{M}_i \leq \hat{M}_i'$.*

Note, that with this definition $\mathbf{M} \preceq \mathbf{M}'$ implies $\Pi^1(\mathbf{M}) \leq \Pi^1(\mathbf{M}')$.

Proposition 11. *Boundedness is decidable for* Ueos *wrt. value semantics.*

Proof. A partial order $\leq$ has the property of strict transitive compatibility (monotonicity), i.e. if $m_1 < m_2$ and $m_1 \xrightarrow{*} m_1'$ then there exists a sequence $m_2 \xrightarrow{*} m_2'$ with $m_1' < m_2'$. Generalising the result of [KM69] it is shown in [FS01], that the boundedness problem is decidable iff $\leq$ is a decidable, strict partial order and the set of successor markings is decidable. Obviously $\preceq$ is decidable and strict transitive compatibility and set the of successors is effective constructible. □

5.3 Relationship to Turing Machines

An even stronger result than the undecidability of reachability is that some of the above mentioned formalism also have the power of Turing machines. This can be shown by reduction to other extended Petri net formalisms which have Turing power or by simulating Turing machines or two-counter-machines directly.

The generalisation of Ueos are minimal OO-nets [Kum00] where the tokens are objects. Objects are characterised by their identity (and maybe several other attributes which are not considered). The only assumption of OO-nets is an countable set of object identities which can be tested for equality and that fresh identities can be created at run-time. This minimal assumption should be met by every existing object-oriented extension of Petri nets. Under this assumptions OO-nets can simulate two-counter-machines. This implies that reachability is also undecidable for OO-nets and therefore for almost every object-oriented extension of Petri nets.

Proposition 12 ([Kum00]). *Minimal OO-Nets are Turing powerful.*

It is easy to see, that the fusion construction of Def. 4 can be extended for an arbitrary but fixed number of nets, i.e. instances with the restriction that there are only finitely many synchronisations. From Proposition 1 we know, that the fusion net can always be simulated by a P/T-net which is not Turing powerful. The conclusion is that the identities create the expressiveness of the model.

For Nested Petri Nets without recursion (this is important) the termination problem is decidable, which is not for Turing machines, so Nested Petri Nets are not Turing powerful. Recursion means, that it is possible to create new net which may results in unbounded hierarchies.

Proposition 13 ([LS00]). *Termination is decidable for recursion-free Nested Petri Nets.*

For the proof of Proposition 13 the restriction of Nested Petri Nets, that no variable appears twice in the input arcs, it is essential, since without this condition the firing rule is not monotonic with is a critical assumption for the termination proof in [FS01] which is used in the proof of Theorem 13 in [LS00]: Assume a transition with the variable x occurring twice in the pre-set (which is forbidden). In a marking m with two net tokens in the pre-set having equal markings the transition is enabled. In the marking m' where one of the two net tokens has a greater marking than the other we have $m' \geq m$ but t is *not* activated, i.e. the firing rule is no longer monotonic.[6]

In the following it it shown that for the "nets within nets" approach undecidability is neither due to the use of infinite sets of identities (as for object-oriented Petri nets) nor due to the creation operation itself (as for Nested Petri Nets) It is the recursive structure that creates the expressiveness, since the "nets within nets" model is Turing powerful even if *no identities* are used and *no recursive creation* is allowed. It is sufficient to allow the operation of vertical synchronisation to simulate Turing tapes or counters.[7]

In the following it it shown that it is the recursive structure that creates the expressiveness, since the "nets within nets" model is Turing powerful even if *no identities* are used. It is sufficient to allow the operation of vertical nesting and synchronisation to simulate Turing machines.

Definition 14. *A* 2-counter automaton *is represented by the tuple*

$$A = (Q, \delta_{0,0}, \delta_{0,1}, \delta_{1,0}, \delta_{1,1}, q_0, Q_f),$$

where Q is a finite set of states, $q_0 \in Q$ is the initial state, and $Q_f \subseteq Q$ is the set of final states. The transitions function $\delta_{0,0}$ to $\delta_{1,1}$ are defined depending on the values of the counters:

[6] For Ueos, Anonymous Object-Net Systems, and derived models like Mobile Object-Net Systems [KR03,Köh03] this problem does not arises since the firing rule used for Ueos or Ons is formulated independently of the inner marking of net tokens.

[7] A similar proof idea exist for the ambient calculus [CGG00] to show that the ambient calculus is Turing complete. Here the nested ambients are used to encode the tape of a Turing machine. The head of the Turing machine itself is also an ambient, which moves around the nested cell-ambients (i.e. the tape).

- $\delta_{0,0} : Q \to Q \times \{1\} \times \{1\}$ *(if both counters are equal zero),*
- $\delta_{0,1} : Q \to Q \times \{1\} \times \{-1, 1\}$ *(if the first counter equals zero),*
- $\delta_{1,0} : Q \to Q \times \{-1, 1\} \times \{1\}$ *(if the second counter equals zero), and*
- $\delta_{1,1} : Q \to Q \times \{-1, 1\} \times \{-1, 1\}$ *(if both counters are positive).*

A configuration *is a triple* $(q, n_1, n_2) \in Q \times \mathbb{N} \times \mathbb{N}$. *The successor configuration of* (q, n_1, n_2) *is* $(q', n_1 + d_1, n_2 + d_2)$ *where* $k = \min(n_1, 1)$, $l = \min(n_2, 1)$, *and* $(q', d_1, d_2) = \delta_{k,l}(q)$. *Note, that* $\delta_{k,l}$ *is actually a test for zero which guarantees* $n_1 + d_1 \geq 0$ *and* $n_2 + d_2 \geq 0$.

The following proof of Prop. 14 encoded natural numbers by exploiting the nesting structure of the net-tokens, like $\mathsf{suc}[\mathsf{suc}[\ldots \mathsf{suc}[\mathsf{zero}]\ldots]]$. Incrementation is implemented by synchronising downwards the marking structure and modifying token zero into $\mathsf{suc}[\mathsf{zero}]$. Similarly for decrementation.

Proposition 14. *Anonymous Object Net Systems are Turing powerful.*

Proof. Let A be a 2-counter machine, then define the object net $N(A) = (P_1 \cup P_2, T_1 \cup T_2, \mathrm{pre}, \mathrm{post}, M_0)$ as follows: Let $P_1 = \{q_1, q_2 \mid q \in Q\}$, where the place q_1 holds the first and q_2 the second counter (encoded as a net-token) of A when being in state q. The places $\{q_1, q_2 \mid q \in Q_f\}$ describe the final states. Let $T_1 = \{\mathsf{f}_{(q,q',k,l)}, \mathsf{j}_{(q,q',k,l)} \mid \delta_{k,l}(q) = (q', d_1, d_2)\}$, where $k = 0$ ($l = 0$) iff the first (second) counter equals zero.
Let $P_2 = \{p_{(q,q',k,l),i}, p'_{(q,q',k,l),i}, p''_{(q,q',k,l),i} \mid i \in \{1, 2\} \wedge \delta_{k,l}(q) = (q', d_1, d_2)\}$ and $T_2 = \{t_{(q,q',k,l),i}, t'_{(q,q',k,l),i} \mid i \in \{1, 2\} \wedge \delta_{k,l}(q) = (q', d_1, d_2)\}$, used to modify the counters. Let $V = \{\mathsf{inc}_i, \mathsf{set}_i, \mathsf{dec}_i, \mathsf{test}_{k,i} \mid i \in \{1, 2\} \wedge k \in \{0, 1\}\}$ be the set of channels. The initial marking is $M_0 = q_{0,1} + q_{0,2}$, i.e. the initial state of A. We use the abbreviation $c := (q, q', k, l)$ for subscripts.

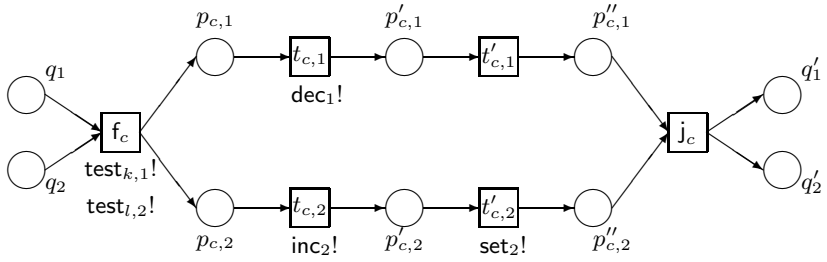

Fig. 7. Simulation of the state change $\delta_{k,l}(q) = (q', -1, 1)$

Define the type $d(q_1) = d(p_{c,1}) = d(p'_{c,1}) = d(p''_{c,1}) = N_{suc,1}$ and $d(q_2) = d(p_{c,2} = d(p'_{c,2}) = d(p''_{c,2}) = N_{suc,2}$. Pre- and post-condition are given as:

- $\mathrm{pre}(\mathsf{f}_c) = q_1 + q_2$, $\mathrm{post}(\mathsf{f}_c) = p_{c,1} + p_{c,2}$, $\lambda^{\uparrow}(\mathsf{f}_c) = 0$, and $\lambda^{\downarrow}(\mathsf{f}_c) = \mathsf{test}_{k,1} + \mathsf{test}_{l,2}$.
- $\mathrm{pre}(\mathsf{j}_c) = p''_{c,1} + p''_{c,2}$, $\mathrm{post}(\mathsf{j}_c) = q'_1 + q'_2$, and $\lambda^{\uparrow}(\mathsf{j}_c) = \lambda^{\downarrow}(\mathsf{j}_c) = 0$
- $\mathrm{pre}(t_{c,i}) = p_{c,i}$, $\mathrm{post}(t_{c,i}) = p'_{c,i}$, $\lambda^{\uparrow}(t_{c,i}) = 0$ and
  $\lambda^{\downarrow}(t_{c,i}) = \begin{cases} \mathsf{inc}_i, & \text{if } d_i = +1 \text{ where } \delta_{k,l}(q) = (q', d_1, d_2) \\ \mathsf{dec}_i & \text{if } d_i = -1 \text{ where } \delta_{k,l}(q) = (q', d_1, d_2) \end{cases}$.

– $\mathrm{pre}(t'_{c,i}) = p'_{c,i}$, $\mathrm{post}(t'_{c,i}) = p''_{c,i}$, $\lambda^{\uparrow}(t'_{c,i}) = 0$, and
$\lambda^{\downarrow}(t'_{c,i}) = \begin{cases} \mathsf{set}_i, & \text{if } d_i = +1 \text{ where } \delta_{k,l}(q) = (q', d_1, d_2) \\ 0, & \text{if } d_i = -1 \text{ where } \delta_{k,l}(q) = (q', d_1, d_2) \end{cases}$.

Figure 7 shows the state change $\delta_{k,l}(q) = (q', -1, 1)$ incrementing the first and decrementing the second counter (where uplinks are denoted with "?" and downlinks with "!").

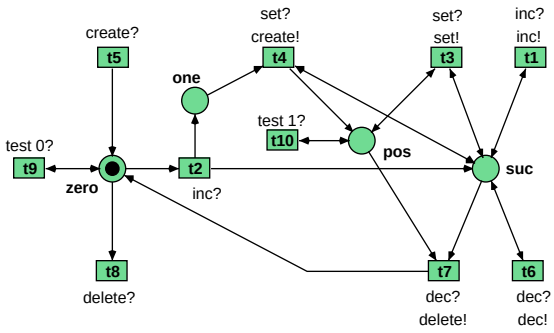

Fig. 8. The "Successor" Net N_{suc}

Figure 8 shows the structure of the successor net $N_{suc,1}$ and $N_{suc,2}$ which is the same (the index i of nodes are omitted in the inscriptions). The net $N_{suc,i}$ implements a successor cell. The type mapping is defined by $d(\mathsf{zero}_i) = d(\mathsf{one}_i) = d(\mathsf{pos}_i) = N_\bullet$ and $d(\mathsf{suc}_i) = N_{suc,i}$.

Define the encoding of the counters using $cod_i(n+1) = \mathsf{pos}_i + \mathsf{suc}_i[cod_i(n)]$ and $cod_i(0) = \mathsf{zero}_i$ (where $i \in \{1, 2\}$ denotes the first or the second counter, resp.). So, the counter n is represented by the nested marking structure:

$$cod_i(n) := \underbrace{\mathsf{suc}_i[\mathsf{pos}_i + \mathsf{suc}_i[\ldots \mathsf{pos}_i + \mathsf{suc}_i[}_{n-\text{times}} \mathsf{zero}_i \underbrace{]\ldots]]}_{n-\text{times}}$$

The object system $ONS(A) = (\mathcal{N}, d, V)$ with $\mathcal{N} = \{N(A), N_{suc,1}, N_{suc,2}, N_\bullet\}$ simulates A. Its initial marking is

$$\mathbf{M}_0 = q_{0,1}[\mathsf{zero}_1] + q_{0,2}[\mathsf{zero}_2] = q_{0,1}[cod_1(0)] + q_{0,2}[cod_2(0)]$$

We prove, that whenever the automaton A changes its configuration (q, n_1, n_2) to $(q', n_1 + d_1, n_2 + d_2)$, then the object system $ONS(A)$ changes from $q_1[cod_1(n_1)] + q_2[cod_2(n_2)]$ to $q'_1[cod_1(n_1 + d_2)] + q'_2[cod_2(n_2 + d_2)]$:

1. Initially, A makes two tests for zero by choosing the appropriate $\delta_{k,l}$. This is simulated by the synchronisation on the channels $\mathsf{test}_{k,1}$ und $\mathsf{test}_{l,2}$ of the transition $\mathsf{f}_{(q,q',k,l)}$.
2. Incrementation (i.e. $d_i = +1$) of $cod(n_i) = \mathsf{suc}_i[\cdots[\mathsf{pos}_i + \mathsf{suc}_i[\mathsf{zero}_i]]\cdots]$ involves two steps: The first one fires

$$\theta_1 = t_{(q,q',k,l),i} /\!\!/_{\mathsf{inc}_i} \underbrace{t_{1,i} /\!\!/_{\mathsf{inc}_i} \cdots /\!\!/_{\mathsf{inc}_i} t_{1,i} /\!\!/_{\mathsf{inc}_1}}_{n\text{-times}} t_{2,i}$$

which is a synchronisation along the channel inc_1. Note, that θ_1 is the only possible transition when starting to synchronise on channel inc_1. This step removes the token from the place zero_i and creates a new cell on the place suc_i with the empty marking (due to the preservation of the marking sum). So, we obtain $\mathsf{suc}_i[\cdots[\mathsf{pos}_i + \mathsf{suc}_i[\mathsf{one}_i + \mathsf{suc}_i[0]]]\cdots]$. The token on the place one_i indicates the net that contains the cell, that still has to be initialised. By firing n-times via the channel set_i and once via create_i:

$$\theta_2 = t'_{(q,q',k,l),i} /\!\!/_{\mathsf{set}_1} \underbrace{t_{3,i} /\!\!/_{\mathsf{set}_1} \cdots /\!\!/_{\mathsf{set}_i} t_{3,i} /\!\!/_{\mathsf{set}_i}}_{n\text{-times}} t_{4,i} /\!\!/_{\mathsf{create}} t_{5,i}$$

The resulting marking is $\mathsf{suc}_i[...[\mathsf{pos}_i + \mathsf{suc}_i[\mathsf{pos}_i + \mathsf{suc}_i[\mathsf{zero}_i]]]...] = cod(n_i+1)$. Note, that θ_2 is the only possible sequence when starting to synchronise on channel set_i. The token on one_i is used to switch the synchronisation sequence from set_i to create_i. The token on pos_i indicates cells describing a counter greater than zero.

3. Decrementation (i.e. $d_i = -1$) of $cod(n_i+1) = \mathsf{suc}_i[...\mathsf{suc}_i[\mathsf{pos}_i + \mathsf{suc}_i[\mathsf{zero}_i]]...]$ is implemented by firing

$$\theta_3 = t_{(q,q',k,l),i} /\!\!/_{\mathsf{dec}_i} \underbrace{t_{6,i} /\!\!/_{\mathsf{dec}_i} \cdots /\!\!/_{\mathsf{dec}_i} t_{6,i} /\!\!/_{\mathsf{dec}_i}}_{n\text{-times}} t_{7,i} /\!\!/_{\mathsf{delete}_i} t_{8,i}$$

which is a synchronisation n-times via the channel dec_i and once via delete_i. Note, that θ_3 is the only possible transition when starting to synchronise on channel dec_i. This step removes the innermost cell and puts one token onto the place zero_i. The resulting marking is $\mathsf{suc}_i[\cdots \mathsf{suc}_i[\mathsf{zero}_i]\cdots] = cod(n_i)$.

So, $ONS(A)$ simulates A. Since 2-counter machines are equivalent to Turing machines the proposition is proven. □

Note, that this technique does not rely on unbounded synchronisation: The synchronisations in Figure 8 can be implemented using paired synchronisation only. So, it is shown that for the "nets within nets" approach Turing completeness can be established even if the operation of vertical synchronisation is limited to span over one nesting level, i.e. with paired synchronisation only.

6 Conclusion

In this contribution we presented decidability results for "nets within nets" formalisms, i.e. formalisms that allow for Petri nets as tokens. The analysis of decidable properties has showed that Ueos are more than just a convenient representation of another – possibly larger – Petri net model – since the reachability problem is undecidable for Ueos. So, Ueos are a real extension with greater computational power. Nevertheless, interesting questions – like boundedness – remain decidable, making Ueos weaker than Petri nets with reset arcs.

Since Kummer showed that Petri net formalism having object identities are Turing powerful the formalism of anonymous Object-Net Systems has been investigated, which allows nets as tokens in a recursive way. It has been shown that

for this minimal extension of Ueos the formalism allowing arbitrary deep nesting Turing power is obtained. So, the investigation of formalisms that are less expressive is worthwhile. An interesting starting point are Ons with bounded nesting depth.

References

[ACR00] G. Agha, F. De Cindio, and G. Rozenberg, editors. *Concurrent Object-Oriented Programming and Petri Nets*, volume 2001 of *Lecture Notes in Computer Science*. Springer-Verlag, 2000.

[AK77] Toshiro Araki and Tadao Kasami. Some decision problems related to the reachability problem for Petri nets. *Theoretical Computer Science*, 3:85–104, 1977.

[BJM00] Adel Bouhoula, Jean-Pierre Jouannaud, and José Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236:35–132, 2000.

[CDE+99] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José Quesada. *Maude: Specification and Programming in Rewriting Logic. Maude System documentation*, 1999.

[CGG99] Luca Cardelli, Andrew D. Gordon, and Giorgio Ghelli. Mobility types for mobile ambients. In *Proceedings of the Conference on Automata, Languages, and Programming (ICALP'99)*, volume 1644 of *Lecture Notes in Computer Science*, pages 230–239. Springer-Verlag, 1999.

[CGG00] Luca Cardelli, Andrew D. Gordon, and Giorgio Ghelli. Ambient groups and mobility types. Technical report, Microsoft Research and University of Pisa, 2000.

[Cia94] Gianfranco Ciardo. Petri nets with marking-dependent arc cardinality: properties and analysis. In Robert Valette, editor, *International Conference on Application and Theory of Petri Nets*, volume 815 of *Lecture Notes in Computer Science*, pages 179–199. Springer-Verlag, 1994.

[DFS98] Catherine Dufourd, Alain Finkel, and Philippe Schnoebelen. Reset nets between decidability and undecidability. In K. Larsen, editor, *Automata, Languages, and Programming (ICALP'98)*, volume 1443 of *Lecture Notes in Computer Science*, pages 103–115. Springer-Verlag, 1998.

[EM85] Hartmut Ehrig and Bernd Mahr. *Fundamentals of algebraic Specification*. EATCS Monographs on TCS. Springer-Verlag, 1985.

[EN94] Javier Esparza and Morgens Nielsen. Decidability issues for Petri nets – a survey. *Journal on Information Processing and Cybernetics*, 30(3):143–160, 1994.

[Far99] Berndt Farwer. A linear logic view of object Petri nets. *Fundamenta Informaticae*, 37(3):225–246, 1999.

[FS01] Alain Finkel and Philippe Schnoebelen. Well-structured transition systems everywhere! *Theoretical Computer Science*, 256(1-2):63–92, 2001.

[HU79] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 1979.

[KF03] Michael Köhler and Berndt Farwer. Mobile object-net systems and their processes. In *Proceedings of the International Workshop on Concurrency, Specification, and Programming, CS&P 2003*, pages 134–149, 2003.

[KM69] Richard M. Karp and Raymond E. Miller. Parallel program schemata. *Journal of Computer and System Sciences*, 3(2):147–195, May 1969.

[KMR03] Michael Köhler, Daniel Moldt, and Heiko Rölke. Modelling mobility and mobile agents using nets within nets. In W. v. d. Aalst and E. Best, editors, *Proceedings of the International Conference on Application and Theory of Petri Nets 2003*, volume 2679 of *Lecture Notes in Computer Science*, pages 121–140. Springer-Verlag, 2003.

[Köh03] Michael Köhler. Mobile object net systems. In *10. Workshop Algorithmen und Werkzeuge für Petrinetze*, pages 51–60. Universität Eichstätt, 2003.

[KR03] Michael Köhler and Heiko Rölke. Concurrency for mobile object net systems. *Fundamenta Informaticae*, 54(2-3), 2003.

[Kum00] Olaf Kummer. Undecidability in object-oriented Petri nets. *Petri Net Newsletter*, 59:18–23, 2000.

[Kum02] Olaf Kummer. *Referenznetze*. Logos Verlag, 2002.

[KWD03] Olaf Kummer, Frank Wienberg, and Michael Duvigneau. *The Reference Net Workshop (Renew)*. Universität Hamburg, 1998-2003. http://www.renew.de.

[Lip76] Richard J. Lipton. The reachability problem requires exponential space. Research Report 62, Dept. of Computer science, 1976.

[Lom00] Irina A. Lomazova. Nested Petri nets – a formalism for specification of multi-agent distributed systems. *Fundamenta Informaticae*, 43(1-4):195–214, 2000.

[LS00] Irina A. Lomazova and Philippe Schnoebelen. Some decidability results for nested Petri nets. In *International Conference on Perspectives of System Informatics (PSI'99)*, volume 1755 of *Lecture Notes in Computer Science*, pages 208–220. Springer-Verlag, 2000.

[May81] Ernst W. Mayr. An algorithm for the general Petri net reachability problem. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 238–246, 1981.

[Mes92] José Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.

[MM90] José Meseguer and Ugo Montanari. Petri nets are monoids. *Information and Computation*, 88(2):105–155, October 1990.

[Rei91] Wolfgang Reisig. Petri nets and algebraic specifications. *Theoretical Computer Science*, 80:1–34, 1991.

[SMÖ01] Mark-Oliver Stehr, José Meseguer, and Peter C. Ölveczky. Rewriting logic as a unifying framework for Petri nets. In H. Ehrig, G. Juhas, J. Padberg, and G. Rozenberg, editors, *Unifying Petri Nets*, Lecture Notes in Computer Science (Advances in Petri Nets). Springer-Verlag, December 2001.

[Val78] Rüdiger Valk. Self-modifying nets, a natural extension of Petri nets. In Ausiello, G. and Böhm, C., editors, *Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 464–476. Springer-Verlag, 1978.

[Val98] Rüdiger Valk. Petri nets as token objects: An introduction to elementary object nets. In Jörg Desel and Manuel Silva, editors, *Application and Theory of Petri Nets*, volume 1420 of *Lecture Notes in Computer Science*, pages 1–25, 1998.

A Petri Net Notations

The definition of Petri nets relies on the notion of multisets. A multiset on the set D is a mapping $A : D \to \mathbb{N}$. The set of all multisets over the set D is denoted $MS(D)$. Multisets are generalisations of sets in the sense that every subset of D corresponds to a multiset A with $A(x) \leq 1$ for all $x \in D$. The empty multiset 0 is defined as $0(x) = 0$ for all $x \in D$.

Any mapping $f : D \to D'$ can be generalised to a mapping $f : MS(D) \to MS(D')$ on multisets: $f(\sum_1^n a_i) = \sum_1^n f(a_i)$. This includes the special case $f(0) = 0$. These definitions are in accordance with the set-theoretic notation $f(A) = \{f(a) \mid a \in A\}$.

We use common notations for the cardinality of a multiset $|A|$, multiset union $A_1 + A_2$, multiset-difference $A_1 - A_2$, and multiset ordering $A_1 \geq A_2$. The notation is overloaded, being used for sets as well as multisets. The meaning will be apparent from its use.

A Petri net system is given by a quadruple, consisting of disjoint sets of places and transitions, together with a flow relation connecting places and transitions, and a simple marking, given by a subset of the places.

Definition 15. *$N = (P, T, F, M_0)$ is a* Petri net *iff the set of places P and the set of transitions T are disjoint, i.e. $P \cap T = \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation, and $M_0 \subseteq P$ is the initial marking.*

Some commonly used notations for Petri nets are ${}^\bullet y := \{x \mid xFy\}$ for the *preset* and $x^\bullet := \{y \mid xFy\}$ for the *postset* of a net element y.

A marking $M \subseteq P$ of a Petri net *enables* a transition t iff ${}^\bullet t \subseteq M$ and $t^\bullet \cap M = \emptyset$ hold, i.e. if the input places each hold a token and all output places do not hold any token. If marking $M \subseteq P$ enables transition t, it can fire and the successor marking $M' = (M \setminus {}^\bullet t) \cup t^\bullet$ is reached.

A P/T-net is an extension of a Petri net, in which the arcs are inscribed by non-negative integers and places can hold more than one token.

Definition 16. *A* P/T-net *N is a tuple $N = (P, T, pre, post, M_0)$, such that: P is a finite set of places. T is a finite set of transitions, with $P \cap T = \emptyset$. $pre, post : T \to MS(P)$ are the pre- and post-condition functions, resp. $M_0 \in MS(P)$ is the initial marking of N.*

A transition $t \in T$ of a P/T-net $N = (P, T, \mathrm{pre}, \mathrm{post}, M_0)$ is enabled in marking M iff $M \geq \mathrm{pre}(t)$ holds. The successor marking when firing t is $M' = M - \mathrm{pre}(t) + \mathrm{post}(t)$. We denote the enablement of t in marking M by $M \xrightarrow[N]{t}$. Firing of t is denoted by $M \xrightarrow[N]{t} M'$.

B Algebraic Specification and Rewrite Systems

An algebraic specification consists of a set of operators, which is interpreted by a concrete algebra (cf. [EM85]). A multi-sorted *signature* $\Sigma = (K, \Omega)$ consists of a finite set of *kinds* K and a finite indexed set $\Omega = \{\Omega_{w,k}\}_{w \in K^*, k \in K}$ of *operators*. An operator $\omega \in \Omega_{\lambda,k}$ with no arguments denotes a constant of kind k. An extension of the multi-sorted case is *Membership Equational Logic* (MEL) [BJM00]. A MEL-signature $\Sigma = (K, \Omega, S)$ is a tuple where (K, Ω) is a multi-sorted signature and S is a K-indexed set of sub-sorts: $S = \{S_k\}_{k \in K}$, with $S_k \cap K = \emptyset$.

Let $\Sigma = (K, \Omega)$ be a signature and $X = \{X_k\}_{k \in K}$ an indexed set of variables with disjoint X_k. The set $\mathbb{T}^k_\Sigma(X)$ denotes the set of all term over the signature using variables from X. The set of ground terms of sort k is $\mathbb{T}^k_\Sigma := \mathbb{T}^k_\Sigma(\emptyset)$. The set $\mathbb{T}_\Sigma(X)$ denotes the set of all term over the signature and $\mathbb{T}_\Sigma$ denotes the set of ground terms.

Let $t \in \mathbb{T}^k_\Sigma(X)$ be a term. The set of variables used in t is denoted $\mathit{Vars}(t)$ defined by $\mathit{Vars}(x) := \{x\}$ for $x \in X_k$ and $\mathit{Vars}(\omega(t_1, \ldots, t_n)) := \bigcup_{i=1}^n \mathit{Vars}(t_i)$.

A *MEL-formulae* ϕ has the form:

1. $M \in s$ for $s \in S_k$ and $M \in \mathbb{T}^k_\Sigma$ (membership).
2. $M = N$ for $M, N \in \mathbb{T}^k_\Sigma$.
3. $\phi_1 \wedge \ldots \wedge \phi_n$ for MEL-formulae ϕ_i.

A *MEL-axiom* has the form $\forall X : \phi \Longrightarrow M \in s$ (membership axiom) or $\forall X : \phi \Longrightarrow M = N$(equational axiom).

A *MEL data-type specification* $\mathcal{D} = (\Sigma, X, E)$ consists of a signature $\Sigma = (K, \Omega, S)$, a disjoint indexed set of variables $X = \{X_k\}_{k \in K}$, and an indexed set $E = \{E_k\}_{k \in K}$ of axioms.

The semantics of a signature is called Σ-algebra and consists of a set of carrier sets and a family of functions over the carrier sets. A Σ-algebra is called a Theory if all axioms of the specification are true for all variable assignments. It is well known that the set of equivalence classes $\mathbb{T}^k_{\Sigma,E}(X)$ wrt. to the axioms E is the initial object in the category of all theories. For details cf. [EM85,BJM00].

Rewriting logic [Mes92] combines *equational* data types with *rewriting* rules. This allows to specify equality and dynamic change. Rewriting logic is executable using for example the Maude tool [CDE+99].

Definition 17. *A* conditional rewriting system *is a tuple* $\mathcal{R} = (\mathcal{D}, L, R)$, *where:*

- $\mathcal{D} = (\Sigma, X, E)$ *is a MEL specification.*
- L *is a set of labels.*
- $R \subseteq L \times (\mathbb{T}^2_{\Sigma,E}(X))^+$ *is a set of rewrite rules.*

A rule $(l, ([t], [t'])([u_1], [v_1]) \ldots ([u_k], [v_k])) \in R$ is denoted using the more suggestive notation:

$$l : [t] \to [t'] \text{ if } [u_1] \to [v_1] \wedge \ldots \wedge [u_k] \to [v_k]$$

The term $[u_1] \to [v_1] \wedge \ldots \wedge [u_k] \to [v_k]$ is the condition of the rule.

Given a rewrite system $\mathcal{R}$ a sequence $\mathcal{R} \vdash [t] \to [t']$ is deducible, iff it can be generated using the following deduction rules:

1. *Reflexivity.* For each $[t] \in \mathbb{T}_{\Sigma,E}(X)$ it holds:

$$\frac{}{[t] \to [t]}$$

2. *Σ-structure.* For each operator $f \in k_1 \times \cdots \times k_n \to k$ and $t_i, t_i' \in \mathbb{T}_{\Sigma,E}^{k_i}(X)$ for $1 \leq i \leq n$ there exists the rule:

$$\frac{[t_1] \to [t_1'] \ldots [t_n] \to [t_n']}{[f(t_1, \ldots, t_n)] \to [f(t_1', \ldots, t_n')]}$$

3. *Replacement.* For each rule

$$r : [t(\bar{x})] \to [t'(\bar{x})] \text{ if } [u_1(\bar{x})] \to [v_1(\bar{x})] \wedge \ldots \wedge [u_k(\bar{x})] \to [v_k(\bar{x})]$$

in $\mathcal{R}$ and for $[w_i], [w_i'] \in \mathbb{T}_{\Sigma,E}^{k_i}(X)$, $x_i \in X_{k_i}$ $(1 \leq i \leq n)$ there exists the rule:

$$\frac{\begin{array}{c}[w_1] \to [w_1'] \ldots [w_n] \to [w_n'] \\ [u_1(\bar{w}/\bar{x})] \to [v_1(\bar{w}/\bar{x})] \ldots [u_k(\bar{w}/\bar{x})] \to [v_k(\bar{w}/\bar{x})]\end{array}}{[t(\bar{w}/\bar{x})] \to [t'(\bar{w}'/\bar{x})]}$$

4. *Transitivity.*

$$\frac{[t_1] \to [t_2] \qquad [t_2] \to [t_3]}{[t_1] \to [t_3]}$$

The semantics of a rewriting system is defined by the category, with equivalence classes of terms as objects and deduction rules as morphism (cf. [Mes92]).

C The Succesor Net in Renew-Syntax

The successor cell given in RENEW-syntax [KWD03] is pictured in Figure 9, a test net exploiting the successor net is shown in Figure 10.

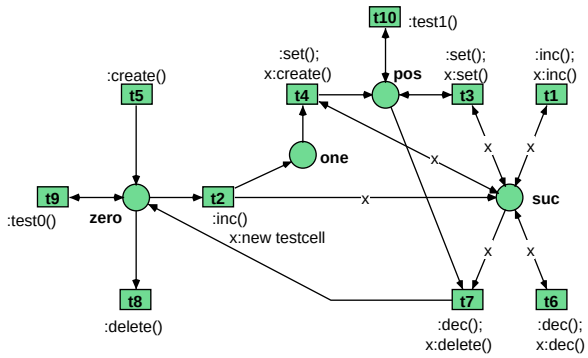

Fig. 9. The Successor Cell

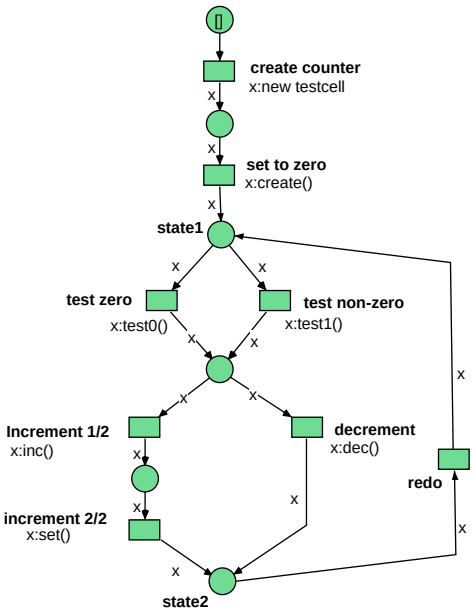

Fig. 10. A Net using a Counter Net