On the Complexity of Safe, Elementary Hornets

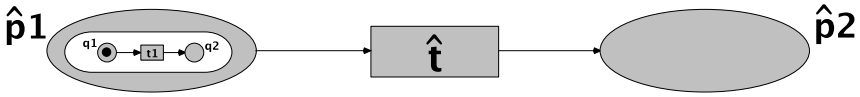
Michael Köhler-Bußmeier

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

Abstract. In this paper we study the complexity of HORNETS, an algebraic extension of object nets. We define a restricted class: safe, elementary HORNETS, to guarantee finite state spaces. It will turn out, that the reachability problem for this class requires exponential space, which is a major increase when compared to safe, elementary object nets, which require polynomial space.
Keywords: Hornets, nets-within-nets, object nets, reachability, safeness

1 Hornets: Higher-Order Object Nets

In this paper we study the algebraic extension of object nets, called HORNETS. HORNETS are a generalisation of object nets [KR03,KR04,KBH09], which follow the *nets-within-nets* paradigm as proposed by Valk [Val91,Val03]. With object nets we study Petri nets where the tokens are nets again, i.e. we have a nested marking. Events are also nested. We have three different kinds of events – as illustrated by the following example:



1. System-autonomous: The system net transition $\widehat{t}$ fires autonomously, which moves the net-token from $\widehat{p}_1$ to $\widehat{p}_2$ without changing its marking.
2. Object-autonomous: The object net fires transition t_1 "moving" the black token from q_1 to q_2. The object net remains at its location $\widehat{p}_1$.
3. Synchronisation: Whenever we add matching synchronisation inscriptions at the system net transition $\widehat{t}$ and the object net transition t_1, then both must fire synchronously: The object net is moved to $\widehat{p}_2$ and the black token moves from q_1 to q_2 inside. Whenever synchronisation is specified, autonomous actions are forbidden.

For HORNETS we extend object-nets with algebraic concepts that allow to modify the structure of the net-tokens as a result of a firing transition. This is a generalisation of the approach of algebraic nets [Rei91], where algebraic data types replace the anonymous black tokens.

Example 1. We consider a HORNET with two workflow nets N_1 and N_2 as tokens – cf. Figure 1. To model a run-time adaption, we combine N_1 and N_2 resulting in the net $N_3 = (N_1 \| N_2)$. This modification is modelled by transition t of the HORNETS in Fig. 1. In a binding α with $x \mapsto N_1$ and $y \mapsto N_2$ the transition t is enabled. Assume that $(x \| y)$ evaluates to N_3 for α. If t fires it removes the two net-tokens from p and q and generates one new net-token on place r. The net-token on r has the structure of N_3 and its marking is obtained as a transfer from the token on v in N_1 and the token on s in N_2 into N_3. This transfer is possible since all the places of N_1 and N_2 are also places in N_3 and tokens can be transferred in the obvious way.

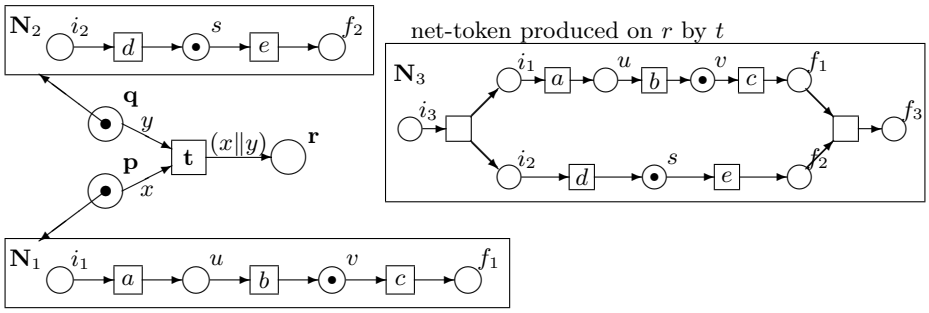


Fig. 1. Modification of the net-token's structure

The use of algebraic operations in HORNETS relates them to *algebraic higher-order (AHO) systems* [HEM05], which are restricted to two-levelled systems but have a greater flexibility for the operations on net-tokens, since each net transformation is allowed. There is also a relationship to Nested Nets [Lom08], which are used for adaptive systems.

It is not hard to prove that the general HORNET formalism is Turing-complete. In [KB09] we have proven that there are several possibilities to simulate counter programs: One could use the nesting to encode counters. Another possibility is to encode counters in the algebraic structure of the net operators.

In this paper we like to study the *complexity* that arises due the algebraic structure. Therefore, we restrict HORNETS to guarantee that the system has a finite state space. First, we allow at most one token on each place, which results in the class of *safe* HORNETS. However this restriction does not guarantee finite state spaces, since we have the nesting depth as a second source of undecidability [KBH09]. Second, we restrict the universe of object nets to finite sets. Finally, we restrict the nesting depth and introduce the class of *elementary* HORNETS, which have a two-levelled nesting structure. This is done in analogy to the class of

*elemtary object net systems (*Eos*)* [KR04], which are the two-level specialisation of general object nets [KR04,KBH09].

If we rule out these sources of complexity the main origin of complexity is the use of algebraic transformations, which are still allowed for safe, elementary Hornets. As a result we obtain the class of safe, elementary Hornets – in analogy to the class of *safe* Eos [KBH10].

We have shown in [KBH10,KBH11] that most problems for *safe* Eos are PSpace-complete. More precisely: All problems that are expressible in LTL or CTL, which includes reachability and liveness, are PSpace-complete. This means that with respect to these problems *safe* Eos are no more complex than p/t nets.

In this presentation we will show that *safe, elementary* Hornets are beyond PSpace. It is well known that "the reachability problem requires exponential space" in the case of *bounded* p/t nets [Lip76]. We obtain that the reachability problem for safe, elementary Hornets requires exponential space, too. We step from polynomial space (as in the case of safe Eos) to exponential space in the case of safe elementary Hornets.

2 Elementary Hornets

A multiset $\mathbf{m}$ on the set D is a mapping $\mathbf{m} : D \to \mathbb{N}$. Multisets can also be represented as a formal sum in the form $\mathbf{m} = \sum_{i=1}^{n} x_i$, where $x_i \in D$.

Multiset addition is defined component-wise: $(\mathbf{m}_1 + \mathbf{m}_2)(d) := \mathbf{m}_1(d) + \mathbf{m}_2(d)$. The empty multiset $\mathbf{0}$ is defined as $\mathbf{0}(d) = 0$ for all $d \in D$. Multiset-difference $\mathbf{m}_1 - \mathbf{m}_2$ is defined by $(\mathbf{m}_1 - \mathbf{m}_2)(d) := \max(\mathbf{m}_1(d) - \mathbf{m}_2(d), 0)$.

The cardinality of a multiset is $|\mathbf{m}| := \sum_{d \in D} \mathbf{m}(d)$. A multiset $\mathbf{m}$ is finite if $|\mathbf{m}| < \infty$. The set of all finite multisets over the set D is denoted $MS(D)$.

Multiset notations are used for sets as well. The meaning will be apparent from its use.

Any mapping $f : D \to D'$ extends to a multiset-homomorphism $f^\sharp : MS(D) \to MS(D')$ by $f^\sharp\left(\sum_{i=1}^{n} x_i\right) = \sum_{i=1}^{n} f(x_i)$.

Net-Algebras We define the algebraic structure of object nets. For a general introduction of algebraic specifications cf. [EM85].

Let K be a set of net-types (kinds). A (many-sorted) *specification* (Σ, X, E) consists of a signature Σ, a family of variables $X = (X_k)_{k \in K}$, and a family of axioms $E = (E_k)_{k \in K}$.

A signature is a disjoint family $\Sigma = (\Sigma_{k_1 \cdots k_n, k})_{k_1, \cdots, k_n, k \in K}$ of operators. The set of terms of type k over a signature Σ and variables X is denoted $\mathbb{T}^k_\Sigma(X)$.

We use (many-sorted) predicate logic, where the terms are generated by a signature Σ and formulae are defined by a family of predicates $\Psi = (\Psi_n)_{n \in \mathbb{N}}$. The set of formulae is denoted PL_Γ, where $\Gamma = (\Sigma, X, E, \Psi)$ is the *logic structure*.

Let Σ be a signature over K. A *net-algebra* assigns to each type $k \in K$ a set $\mathcal{N}_k$ of object nets. Each object $N \in \mathcal{N}_k, k \in K$ net is a p/t net $N = (P_N, T_N, \mathbf{pre}_N, \mathbf{post}_N)$. We identify $\mathcal{N}$ with $\bigcup_{k \in K} \mathcal{N}_k$ in the following. We assume the family $\mathcal{N} = (\mathcal{N}_k)_{k \in K}$ to be disjoint.

The nodes of the object nets in $\mathcal{N}_k$ are not disjoint, since the firing rule allows to transfer tokens between net tokens within the same set $\mathcal{N}_k$. Such a transfer is possible, if we assume that all nets $N \in \mathcal{N}_k$ have the same set of places P_k. In the example of Fig. 1 the object nets N_1, N_2, and N_3 must belong to the same type since otherwise it would be impossible to transfer the markings in N_1 and N_2 to the generated N_3.

In general, P_k is not finite. Since we like each object net to be finite in some sense, we require that the transitions T_N of each $N \in \mathcal{N}_k$ use only a finite subset of P_k, i.e. $\forall N \in \mathcal{N} : |{}^\bullet T_N \cup T_N{}^\bullet| < \infty$.

The family of object nets $\mathcal{N}$ is the universe of the algebra. A net-algebra $(\mathcal{N}, \mathcal{I})$ assigns to each constant $\sigma \in \Sigma_{\lambda,k}$ an object net $\sigma^{\mathcal{I}} \in \mathcal{N}_k$ and to each operator $\sigma \in \Sigma_{k_1 \cdots k_n, k}$ with $n > 0$ a mapping $\sigma^{\mathcal{I}} : (\mathcal{N}_{k_1} \times \cdots \times \mathcal{N}_{k_n}) \to \mathcal{N}_k$.

A net-algebra is called *finite* if P_k is a finite set for each $k \in K$.

A variable assignment $\alpha = (\alpha_k : X_k \to \mathcal{N}_k)_{k \in K}$ maps each variable onto an element of the algebra. For a variable assignment α the evaluation of a term $t \in \mathbb{T}^k_\Sigma(X)$ is uniquely defined and will be denoted as $\alpha(t)$.

A net-algebra, such that all axioms of (Σ, X, E) are valid, is called *net-theory*.

System Net Assume we have a fixed logic $\Gamma = (\Sigma, X, E, \Psi)$ and a net-theory $(\mathcal{N}, \mathcal{I})$. An *elementary higher-order object net* (HORNET) is composed of a system net $\widehat{N}$ and the set of object nets $\mathcal{N}$. W.l.o.g. we assume $\widehat{N} \notin \mathcal{N}$. To guarantee finite state spaces for elementary HORNETS, we require that the net-theory $(\mathcal{N}, \mathcal{I})$ is finite, i.e. each place universe P_k is finite.

The system net is a net $\widehat{N} = (\widehat{P}, \widehat{T}, \mathbf{pre}, \mathbf{post}, \widehat{G})$, where each arc is labelled with a multiset of terms: $\mathbf{pre}, \mathbf{post} : \widehat{T} \to (\widehat{P} \to MS(\mathbb{T}_\Sigma(X)))$. Each transition is labelled by a guard predicate $\widehat{G} : \widehat{T} \to PL_\Gamma$. The places of the system net are typed by the function $d : \widehat{P} \to K$. As a typing constraint we have that each arc inscription has to be a multiset of term of kind assigned to the arc's place:

$$\mathbf{pre}(\widehat{t})(\widehat{p}), \quad \mathbf{post}(\widehat{t})(\widehat{p}) \in MS(\mathbb{T}^{d(\widehat{p})}_\Sigma(X)) \tag{1}$$

For each variable binding α we obtain the evaluated functions $\mathbf{pre}_\alpha, \mathbf{post}_\alpha : \widehat{T} \to (\widehat{P} \to MS(\mathcal{N}))$ in the obvious way.

Nested Markings Since the tokens of an elementary HORNETS are instances of object nets a *marking* is a *nested* multiset of the form:

$$\mu = \sum_{i=1}^{n} \widehat{p}_i[N_i, M_i] \quad \text{where} \quad \widehat{p}_i \in \widehat{P}, N_i \in \mathcal{N}_{d(\widehat{p}_i)}, M_i \in MS(P_{N_i}), n \in \mathbb{N}$$

Each addend $\widehat{p}_i[N_i, M_i]$ denotes a net-token on the place $\widehat{p}_i$ that has the structure of the object net N_i and the marking $M_i \in MS(P_{N_i})$.[1] The set of all nested multisets is denoted as $\mathcal{M}$. We define the partial order $\sqsubseteq$ on nested multisets by setting $\mu_1 \sqsubseteq \mu_2$ iff $\exists \mu : \mu_2 = \mu_1 + \mu$.

[1] For EOS the net structure of a net-token is uniquely determined by the place. So, we do not include the object net N_i in μ for EOS.

The projection Π^1 abstracts from the substructure of all net-tokens:

$$\Pi^1_N\left(\sum\nolimits_{i=1}^{n} \widehat{p}_i[N_i, M_i]\right) := \sum\nolimits_{i=1}^{n} \mathbf{1}_N(N_i) \cdot \widehat{p}_i \tag{2}$$

where the indicator function $\mathbf{1}_N$ is defined as: $\mathbf{1}_N(N_i) = 1$ iff $N_i = N$.

The projection $\Pi^2_{\mathcal{N}}(\mu)$ is the multiset of all object nets in the marking:

$$\Pi^2_{\mathcal{N}}\left(\sum\nolimits_{i=1}^{n} \widehat{p}_i[N_i, M_i]\right) := \sum\nolimits_{i=1}^{n} N_i \tag{3}$$

The projection Π^2_k is the sum of all net-tokens' markings (of the same type $k \in K$) ignoring their local distribution within the system net:

$$\Pi^2_k\left(\sum\nolimits_{i=1}^{n} \widehat{p}_i[N_i, M_i]\right) := \sum\nolimits_{i=1}^{n} \mathbf{1}_k(\widehat{p}_i) \cdot M_i \tag{4}$$

where the indicator function $\mathbf{1}_k$ is defined as: $\mathbf{1}_k(\widehat{p}_i) = 1$ iff $d(\widehat{p}) = k$.

Synchronisation The transitions in an HORNET are labelled with synchronisation inscriptions. We assume a fixed set of channels $C = (C_k)_{k \in K}$.

- The function family $\widehat{l}_\alpha = (\widehat{l}^k_\alpha)_{k \in K}$ defines the synchronisation constraints. Each transition of the system net is labelled with a multiset $\widehat{l}^k(\widehat{t}) = (e_1, c_1) + \cdots + (e_n, c_n)$, where the expression $e_i \in \mathbb{T}^k_\Sigma(X)$ describes the called object net and $c_i \in C_k$ is a channel. Each variable assignment α generates the function $\widehat{l}^k_\alpha(\widehat{t})(N)$:

$$\widehat{l}^k_\alpha(\widehat{t})(N) := \sum\nolimits_{\substack{1 \le i \le n \\ \alpha(e_i) = N}} c_i \quad \text{for} \quad \widehat{l}^k(\widehat{t}) = \sum\nolimits_{1 \le i \le n} (e_i, c_i) \tag{5}$$

  Each function $\widehat{l}^k_\alpha(\widehat{t})(N)$ assigns for each object net N a multiset of channels. The intention is that $\widehat{t}$ fires synchronously with a multiset $\vartheta(N)$ of object net transitions with matching multiset of labels.
- For each $N \in \mathcal{N}_k$ the function l_N assigns to each transition $t \in T_N$ either a channel $c \in C_k$ or $\bot_k$, whenever t fires without synchronisation, i.e. autonously.

Definition 1 (Elementary Hornet). *Assume a fixed many-sorted predicate logic $\Gamma = (\Sigma, X, E, \Psi)$. An elementary* HORNET *is a tuple $EH = (\widehat{N}, \mathcal{N}, \mathcal{I}, d, l, \mu_0)$ such that:*

1. *$\widehat{N}$ is an algebraic net, called the* system net*.*
2. *$(\mathcal{N}, \mathcal{I})$ is a finite net-theory for the logic Γ.*
3. *$d : \widehat{P} \to K$ is the typing of the system net places.*
4. *$l = (\widehat{l}, l_N)_{N \in \mathcal{N}}$ is the labelling.*
5. *$\mu_0 \in \mathcal{M}$ is the initial marking.*

Events The synchronisation labelling generates the set of system events Θ:
The labelling introduces three cases of events:

1. Synchronised firing: There is at least one object net that has to be synchronised, i.e. there is a N such that $\widehat{l}(\widehat{t})(N)$ is not empty.
   Such an event is a pair $\theta = \widehat{t}^{\alpha}[\vartheta]$, where $\widehat{t}$ is a system net transition, α is a variable binding, and ϑ is a function that maps each object net to a multiset of its transitions, i.e. $\vartheta \in MS(T_N)$. It is required that $\widehat{\tau}$ and $\vartheta(N)$ have matching multisets of labels, i.e. $\widehat{l}(\widehat{t})(N) = l_N^{\sharp}(\vartheta(N))$ for all $N \in \mathcal{N}$.
   The intended meaning is that $\widehat{\tau}$ fires synchronously with all the object net transitions $\vartheta(N), N \in \mathcal{N}$.
2. System-autonomous firing: The transition $\widehat{t}$ of the system net fires autonomously, whenever $\widehat{l}(\widehat{t})$ is the empty multiset $\mathbf{0}$.
   We consider system-autonomous firing as a special case of synchronised firing generated by the function ϑ_{id}, defined as $\vartheta_{id}(N) = \mathbf{0}$ for all $N \in \mathcal{N}$.
3. Object-autonomous firing: An object net transition t in N fires autonomously, whenever $l_N(t) = \bot_k$.
   Object-autonomous events are denoted as $id_{\widehat{p},N}[\vartheta_t]$, where $\vartheta_t(N') = \{t\}$ if $N = N'$ and $\mathbf{0}$ otherwise. The meaning is that in object net N fires t autonomously within the place $\widehat{p}$.
   For the sake of uniformity we define for an arbitrary binding α:

$$\mathbf{pre}_{\alpha}(id_{\widehat{p},N})(\widehat{p}')(N') = \mathbf{post}_{\alpha}(id_{\widehat{p},N})(\widehat{p}')(N') = \begin{cases} 1 & \text{if } \widehat{p}' = \widehat{p} \wedge N' = N \\ 0 & \text{otherwise.} \end{cases}$$

The set of all events generated by the labelling l is $\Theta_l := \Theta_1 \cup \Theta_1$:

$$\begin{aligned} \Theta_1 &:= \left\{ \widehat{\tau}^{\alpha}[\vartheta] \quad \mid \forall N \in \mathcal{N} : \widehat{l_{\alpha}}(\widehat{t})(N) = l_N^{\sharp}(\vartheta(N)) \right\} \\ \Theta_2 &:= \left\{ id_{\widehat{p},N}[\vartheta_t] \mid \widehat{p} \in \widehat{P}, N \in \mathcal{N}_{d(\widehat{p})}, t \in T_N \right\} \end{aligned} \tag{6}$$

Firing Rule A system event $\theta = \widehat{\tau}^{\alpha}[\vartheta]$ removes net-tokens together with their individual internal markings. Firing the event replaces a nested multiset $\lambda \in \mathcal{M}$ that is part of the current marking μ, i.e. $\lambda \sqsubseteq \mu$, by the nested multiset ρ. The enabling condition is expressed by the *enabling predicate* ϕ_{EH} (or just ϕ whenever EH is clear from the context):

$$\begin{aligned} \phi(\widehat{\tau}^{\alpha}[\vartheta], \lambda, \rho) \iff & \forall k \in K : \\ & \forall \widehat{p} \in d^{-1}(k) : \forall N \in \mathcal{N}_k : \Pi_N^1(\lambda)(\widehat{p}) = \mathbf{pre}_{\alpha}(\widehat{\tau})(\widehat{p})(N) \wedge \\ & \forall \widehat{p} \in d^{-1}(k) : \forall N \in \mathcal{N}_k : \Pi_N^1(\rho)(\widehat{p}) = \mathbf{post}_{\alpha}(\widehat{\tau})(\widehat{p})(N) \wedge \\ & \Pi_k^2(\lambda) \geq \textstyle\sum_{N \in \mathcal{N}_k} \mathbf{pre}_N^{\sharp}(\vartheta(N)) \wedge \\ & \Pi_k^2(\rho) = \Pi_k^2(\lambda) + \textstyle\sum_{N \in \mathcal{N}_k} \mathbf{post}_N^{\sharp}(\vartheta(N)) - \mathbf{pre}_N^{\sharp}(\vartheta(N)) \end{aligned} \tag{7}$$

The predicate ϕ has the following meaning: Conjunct (1) states that the removed sub-marking λ contains on $\widehat{p}$ the right number of net-tokens, that are removed by $\widehat{\tau}$. Conjunct (2) states that generated sub-marking ρ contains on $\widehat{p}$

the right number of net-tokens, that are generated by $\widehat{\tau}$. Conjunct (3) states that the sub-marking λ enables all synchronised transitions $\vartheta(N)$ in the object N. Conjunct (4) states that the marking of each object net N is changed according to the firing of the synchronised transitions $\vartheta(N)$.

Note, that conjunct (1) and (2) assures that only net-tokens relevant for the firing are included in λ and ρ. Conditions (3) and (4) allow for additonal tokens in the net-tokens.

For system-autonomous events $\widehat{t}^{\alpha}[\vartheta_{id}]$ the enabling predicate ϕ can be simplified further: Conjunct (3) is always true since $\mathbf{pre}_N(\vartheta_{id}(N)) = \mathbf{0}$. Conjunct (4) simplifies to $\Pi^2_k(\rho) = \Pi^2_k(\lambda)$, which means that no token of the object nets get lost when a system-autonomous events fires.

Analogously, for an object-autonomous event $\widehat{\tau}[\vartheta_t]$ we have an idle-transition $\widehat{\tau} = id_{\widehat{p},N}$ and $\vartheta = \vartheta_t$ for some t. Conjunct (1) and (2) simplify to $\Pi^1_{N'}(\lambda) = \widehat{p} = \Pi^1_{N'}(\rho)$ for $N' = N$ and to $\Pi^1_{N'}(\lambda) = \mathbf{0} = \Pi^1_{N'}(\rho)$ otherwise. This means that $\lambda = \widehat{p}[M]$, M enables t, and $\rho = \widehat{p}[M - \mathbf{pre}_N(\widehat{t}) + \mathbf{post}_N(\widehat{t})]$.

Definition 2 (Firing Rule). *Let EH be an elementary* HORNET *and* $\mu, \mu' \in \mathcal{M}$ *markings.*

- *The event* $\widehat{\tau}^{\alpha}[\vartheta]$ *is enabled in* μ *for the mode* $(\lambda, \rho) \in \mathcal{M}^2$ *iff* $\lambda \sqsubseteq \mu \wedge \phi(\widehat{\tau}[\vartheta], \lambda, \rho)$ *holds and the guard* $\widehat{G}(t)$ *holds, i.e.* $E \models^{\alpha}_{\mathcal{I}} \widehat{G}(\widehat{\tau})$.
- *An event* $\widehat{\tau}^{\alpha}[\vartheta]$ *that is enabled in* μ *can fire:* $\mu \xrightarrow[EH]{\widehat{\tau}^{\alpha}[\vartheta](\lambda,\rho)} \mu'$.
- *The resulting successor marking is defined as* $\mu' = \mu - \lambda + \rho$.

Note, that the firing rule has no a-priori decision how to distribute the marking on the generated net-tokens. Therefore we need the mode (λ, ρ) to formulate the firing of $\widehat{\tau}^{\alpha}[\vartheta]$ in a functional way.

3 Boundedness for Safe Elementary Hornets

A HORNET is safe iff each place $\widehat{p}$ in the system net carries at most one token and each net-token carries at most one token on each place p in all reachable markings μ. Since we are interested in safe Hornets in the following, we do not use arc weights greater than 1.

Since all nets $N \in \mathcal{N}_k$ have the same set of places P_k, there is an upper bound for the cardinality of $\mathcal{N}_k$.

Lemma 1. *For each* $k \in K$ *we have* $|\mathcal{N}_k| \leq 2^{\left(2^{4|P_k|}\right)}$.

Proof. Note, that each common set of places P_k is finite for elementary HORNETS. Each possible transition t chooses a subset of P_k for the preset ${}^\bullet t$ and another subset for the postset $t^\bullet$. We identify t with the pair $({}^\bullet t, t^\bullet)$. A transition is an element from $T_k := \mathcal{P}(P_k) \times \mathcal{P}(P_k)$. We have $|T_k| = 2^{|P_k|} \cdot 2^{|P_k|} = (2^{|P_k|})^2 = 2^{2|P_k|}$ different transitions.

To each transition $t \in T_N$, $N \in \mathcal{N}_k$ the partial function l_N assigns either a channel $c \in C_k$ or $\perp_k$.

The set of labelled transitions is $LT_k := T_k \times (C_k \cup \{\bot_k\}))$ and we have $|LT_k| = |T_k \times (C_k \cup \{\bot_k\}))| = 2^{2|P_k|} \cdot (|C_k| + 1)$ different labelled transitions.

We cannot use more channels than we have transitions in the object net, i.e. we could use at most $|T_k| \leq 2^{2|P_k|}$ different channels from $C_k \cup \{\bot_k\}$. Thus, we have:

$$|LT_k| = 2^{2|P_k|} \cdot (|C_k| + 1) \leq 2^{2|P_k|} \cdot 2^{2|P_k|} \leq 2^{4|P_k|}$$

Since each object net N in $\mathcal{N}_k$ is characterised by its set of labelled transitions and there are $|\mathcal{P}(LT_k)| = 2^{|LT_k|}$ subsets of LT_k, we have at most $2^{\left(2^{4|P_k|}\right)}$ different object nets. qed.

In the following we identify an object net with a subset of LT_k.

Lemma 2. *A safe* Hornet *has a finite reachability set.*

Proof. Each reachable marking is of the form $\mu = \sum_{i=1}^{n} \widehat{p}_i[N_i, M_i]$. Since each each object net $N \in \mathcal{N}_k$ is safe, we know that each net-token has at most $2^{|P_k|}$ different markings M_i Furthermore, we know $|\mathcal{N}_k| \leq 2^{\left(2^{4|P_k|}\right)}$. Assume that m is the maximum of all $|P_k|$. Then we have at most $2^{\left(2^{4m}\right)} \cdot 2^m$ different net-tokens $[N, M]$.

Each system net place $\widehat{p}$ is either unmarked or marked with one of these net-tokens. Therefore, we have at most $\left(1 + 2^{\left(2^{4m}\right)} \cdot 2^m\right)^{|\widehat{P}|}$ different markings in the safe Hornet. qed.

It can be shown that for each elementary Hornet EH there exists an Eos $OS(EH)$ with the simulation property: $\mu \xrightarrow[EH]{*} \mu' \iff \overline{\mu} \xrightarrow[OS(EH)]{*} \overline{\mu'}$. The construction of $OS(EH)$ results in a quite drastic increase in the size of Eos, since we add new elements for each object net in the Hornet. By Lemma 1 the size of the Eos grows double exponentially in P_k. Therefore, the simulation by $OS(EH)$ is only good for showing undecidability by giving a reduction from Hornets to Eos (Since we know that the reachability problem for Eos is undecidable [Köh07], we obtain another undecidability proof for Hornets.), but not for studying complexity bounds.

In the following we study the complexity of the reachability problem for safe elementary Hornets.

4 Encoding of Elementary Hornets

To encode a p/t net, we encode the pre-and the post-matrix, which can be stored in $O(|P| \cdot |T|)$ bits. The size of an elementary Hornet is more involved, since we have to store the algebra and the net-tokens:

Lemma 3. *Let* $EH = (\widehat{N}, \mathcal{N}, \mathcal{I}, d, l, \mu_0)$ *an elementary* Hornet *with* $\mu_0 = \sum_{i=1}^{n} \widehat{p}_i[N_i, M_i]$. *Assume that the net algebra can be encoded in* $s(\mathcal{I})$ *space. Then,* EH *can be encoded in* $s(EH) := s(\widehat{N}) + s(\mathcal{I}) + s(\mu_0)$ *space, where:*

$$s(\widehat{N}) \in O\left(|\widehat{P}| \cdot |\widehat{T}|\right) \quad \textit{and} \quad s(\mu_0) \in O\left(|\widehat{P}| \cdot (m_{\mathcal{N}}(\mu_0) \cdot m_P + m_P^2)\right)$$

with $m_P := \max\{|P_k| : k \in K\}$ *and* $m_{\mathcal{N}}(\mu_0) := \max\{s(N) \mid N \in \Pi^2_{\mathcal{N}}(\mu_0)\}$.

Proof. Note, that it is sufficient to encode the system net $\widehat{N}$, the net operators in $\mathcal{I}$, the typing d, the global object net place sets P_k, and the initial marking μ_0. It is not necessary to encode the object nets in $\mathcal{N}$, since the object nets are part of the current marking μ and combinatorially derived from P_k.

- The system net can be stored in $O(|\widehat{P}| \cdot |\widehat{T}|)$ bits:
  The arc-inscriptions $\mathbf{pre}(\widehat{t})$, $\mathbf{post}(\widehat{t})$ and transition-inscriptions $\widehat{G}(\widehat{t})$ are finite and need $O(|\widehat{P}| \cdot |\widehat{T}|)$ bits. The system net labelling $\widehat{l}^k(\widehat{t})$ are finite expressions, too, and need $O(|\widehat{T}|)$ bits.
- The typing d is stored in $O(|\widehat{P}|)$ bits.

We encode the initial marking $\mu_0 = \sum_{i=1}^{n} \widehat{p}_i[N_i, M_i]$ as a list:

- Each $\widehat{p}$ is marked with at most one net-token: $n \leq |\widehat{P}|$.
- Each object net transition is an element from $T_k := \mathcal{P}(P_k) \times \mathcal{P}(P_k)$, which needs $O(|P_k|)$ bits. The label $l_N(t)$ of an object net transition t is stored in $O(|P_k|)$ bits, since $|C_k \cup \{\perp_k\}| \leq |T_N|$.
  Therefore, a labelled object net transition t uses $s(t) \in O(|P_k|)$ bits.
- Each object net is identified with some subset $LT_N = \{(t_1, c_1), \ldots, (t_n, c_n)\}$ of LT_k. We encode each object N as the list $code(N) := (t_1, c_1) \cdots (t_n, c_n)$. All in all, we need $s(N_i) := |LT_{N_i}| \cdot O(|P_k|)$ bits to store $code(N)$.
  Since each place P_k in the object net is marked with at most one token, we can store M_i as a list of those places that are marked in M_i. We need at most $s(M_i) := |P_{d(\widehat{p}_i)}| \log(|P_{d(\widehat{p}_i)}|) \leq |P_{d(\widehat{p}_i)}|^2$ bits for this list.
  Therefore, each net-token $[N_i, M_i]$ needs $O(|LT_{N_i}| \cdot |P_k| + |P_k|^2)$ bits.
- The space needed for the initial marking $\mu_0 = \sum_{i=1}^{n} \widehat{p}_i[N_i, M_i]$ is:

$$\begin{aligned} s(\mu_0) = \textstyle\sum_{i=1}^{n} s(N_i) + s(M_i) \qquad & \in O(\textstyle\sum_{i=1}^{n} |LT_{N_i}| \cdot |P_{d(\widehat{p}_i)}| + |P_k|^2) \\ \subseteq O(\textstyle\sum_{i=1}^{n} m_{\mathcal{N}}(\mu_0) \cdot m_P + m_P^2) & \subseteq O(|\widehat{P}| \cdot (m_{\mathcal{N}}(\mu_0) \cdot m_P + m_P^2)) \end{aligned}$$

Then, $s(EH)$ is the sum of all components. qed.

Since $s(N_i) \leq 2^{4m_P}$ we have $m_{\mathcal{N}}(\mu_0) \leq 2^{4m_P}$, which "simplifies" the estimation for $|\mu_0|$:

$$s(\mu_0) \in O\left(|\widehat{P}| \cdot (m_{\mathcal{N}}(\mu_0) \cdot m_P + m_P^2)\right) \subseteq O\left(|\widehat{P}| \cdot (2^{2(m_P+1)} \cdot m_P + m_P^2)\right)$$

But, in general 2^{4m_P} is a rather rude estimation, e.g. in those cases, when only a small subset of $\mathcal{N}$ is reachable.

5 Complexity of the Reachability Problem

The size $|A|$ of a Turing machines A is the size needed to store its Turing table. Following the presentation in [Esp98] we consider Turing machines A that start

with an empty tape as input. We say that A is exponentially bounded (w.r.t. the size of A) if A uses at most $2^{|A|}$ cells of the tape during all possible runs starting with the empty tape.

It is well known that each TM A can be simulated by a counter program $C(A)$. Moreover, if A is exponentially bounded (w.r.t. the size of A), then each counter of $C(A)$ is bounded by $2^{2^{|A|}}$.

Theorem 1. *The reachability problem for safe elementary* HORNETS *requires exponential space.*

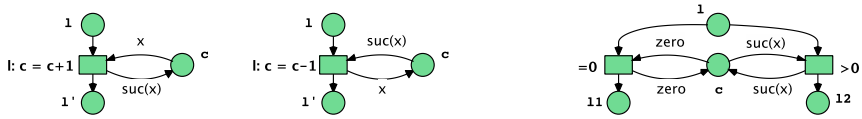
Proof. Our proof that the reachability problem for safe elementary Hornets requires at least exponential space is similar to the heart of Lipton's famous result. Lipton has proven that acceptance of a counter program C translates into a reachability question for a p/t net $N(C)$. Lipton gives a construction for $N(C)$, which is of size $O(|C|^2)$. This proves that the reachability problem requires at least $2^{\sqrt{|C|}}$ space, i.e. exponential space.

Our goal is to simulate a counter program C by a safe, elementary HORNET EH of size $O(poly(|C|))$. In this case we know that a question about a counter program C translates into a question about EH. When C needs $2^{|A|}$ space for the counters, we have that EH needs $2^{\sqrt[n]{|A|}}$ space for the corresponding question, where n is the order of the polynom $poly(|C|)$.

The translation of counter program statements into HORNETS is quite straightforward. For each program line l and each counter c we have a place. Places for lines are marked with black tokens, places for counters are marked with the net-type Int.

For the type Int we have the constant zero and the unary operator suc. The net zero is interpreted as the object net with no transitions and suc enumerates the next subset of LT_{Int}.

The translation of each command into a transition is given as in the figure below. The initial marking puts a black token at l_1 for first command and one net-token $[\mathsf{zero}, \mathbf{0}]$ on each counter place.



l: c = c+1 l: c = c−1 l: if c=0 jmp l1 else jmp l2

Since we cannot use more counters than we have statements in C we have $O(|\widehat{P}|\cdot|\widehat{T}|) \subseteq O(|\widehat{T}|^2) = O(|C|^2)$, i.e. the resulting system net $\widehat{N}$ grows quadratic in the size of the program C.

Each net-token $[\mathsf{zero}, \mathbf{0}]$ in μ_0 needs only $O(|P_{\mathsf{Int}}|)$ bits, since zero is encoded by the empty list. The marking $\mathbf{0}$ is encoded by constantly many bits. Therefore, the initial marking needs only $O(|\widehat{P}|) \subseteq O(|C|)$ bits.

It remains to estimate the size of a Turing machine that the nets zero and $\mathsf{suc}(N)$ (cf. Fig. 2). It is easy to construct the object net zero: simply return the empty list. For the construction of $\mathsf{suc}(N)$, where N is given as a list X, we do

```
function suc(X: list of bit-vector):
                                list of bit-vector
  var a, b, c : bit;
  var i : bit-vector:
  var Y : list of bit-vector:
begin
  Y := empty-list;    c := 1;
  n := d-exp(|P_Int|) // n = 2^(2^(4|P_Int|))
  for i := 0 to n − 1 do
      a := is-in(i, X);
      (b, c) := (c ∧ a, c XOR a);
      if b = 1 then Y := append(i, Y)
  end;
  return Y;
end
```

Fig. 2. Listing: Net Operators

```
function d-exp(n : int) : int
    var x, y, i : int;
begin
    n := 2n + 1;
    x := 1;
    for i := 1 to n do x := 2x end
    // x = 2^(2n+1)
    y = 1;
    for i := 1 to x do y := 2y end
    // y = 2^(2^(2n+1))
    return y
end
```

Fig. 3. Listing: Bound

a binary increment: We convert the list X of labelled transitions into a binary string $a_0 \cdots a_{n-1}$ of length $n := 2^{2^{4|P_{\mathsf{Int}}|}} = |\mathcal{N}_{\mathsf{Int}}|$. The digits a_i are computed whenever necessary. We iterate through all indices i from 0 to n and compute from a and the current carry c the output digit b_i and the next carry bit. We add the labelled transition with index i to the list Y. Finally, Y is the list that encodes the resulting object net $\mathsf{suc}(N)$.

Note, that we cannot denote $n = 2^{2^{4|P_{\mathsf{Int}}|}}$ directly as the limite of the for-loop, since we need at least $2^{4|P_{\mathsf{Int}}|}$ bits to store the for-command, which is undesired to obtain a small Hornet. Instead, we compute n by the function $\mathsf{d\text{-}exp}(|P_{\mathsf{Int}}|)$, which needs only $\log(|P_{\mathsf{Int}}|)$ bits to store. With this trick the size of the programs are within $O(\log(|P_{\mathsf{Int}}|))$.

Each counter of $C(A)$ is bounded by $2^{2^{|A|}}$. If we set $|P_{\mathsf{Int}}| := |A|$, then we can generate $n = 2^{2^{4|A|}} \geq 2^{2^{|A|}}$ object nets – enough to simulate the counter. With $|P_{\mathsf{Int}}| := |A|$ we obtain $O(\log(|P_{\mathsf{Int}}|)) \subseteq O(|P_{\mathsf{Int}}|) = O(|A|) = O(|C|)$.

The size of the HORNET is therefore in

$$O(|\widehat{P}| \cdot |\widehat{T}|) + O(|\widehat{P}|) + O(\log(|P_{\mathsf{Int}}|)) \subseteq O(|C|^2) + O(|C|) + O(\log(|C|)) \subseteq O(|C|^2).$$

This shows that EH needs $2^{\sqrt{|A|}}$ space for the reachability problem. qed.

6 Conclusion

In this paper we studied the subclass of safe, elementary HORNETS. While all properties expressible in temporal logic are PSPACE-complete for safe EOS, we obtain that the reachability problem for safe, elementary HORNETS needs at least exponential space, which is also a lower bound for LTL model checking since the reachability problem can be expressed as a LTL property.

We like to emphasise that the power of safe, elementary HORNETS is only due to the transformations of the net-algebra, since all other sources of complexity (nesting, multiple tokens on a place, etc.) have been ruled out by the restriction of safeness and elementariness.

References

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

[Esp98] Javier Esparza. Decidability and complexity of petri net problems – an introduction. In Wolfgang Reisig and Grzegorz Rozenberg, editors, *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets*, volume 1491 of LNCS, pages 374–428. 1998.

[HEM05] Kathrin Hoffmann, Hartmut Ehrig, and Till Mossakowski. High-level nets with nets and rules as tokens. In *PETRI NETS 2005*, volume 3536 of LNCS, pages 268 – 288. 2005.

[KB09] Michael Köhler-Bußmeier. Hornets: Nets within nets combined with net algebra. In Karsten Wolf and Giuliana Franceschinis, editors, *PETRI NETS 2009*, volume 5606 of LNCS, pages 243–262. 2009.

[KBH09] Michael Köhler-Bußmeier and Frank Heitmann. On the expressiveness of communication channels for object nets. *Fundamenta Informaticae*, 93(1-3):205–219, 2009.

[KBH10] Michael Köhler-Bußmeier and Frank Heitmann. Safeness for object nets. *Fundamenta Informaticae*, 101(1-2):29–43, 2010.

[KBH11] Michael Köhler-Bußmeier and Frank Heitmann. Liveness of safe object nets. *Fundamenta Informaticae*, 112(1):73–87, 2011.

[Köh07] Michael Köhler. Reachable markings of object Petri nets. *Fundamenta Informaticae*, 79(3-4):401 – 413, 2007.

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

[KR04] Michael Köhler and Heiko Rölke. Properties of Object Petri Nets. In J. Cortadella and W. Reisig, editors, *PETRI NETS 2004*, volume 3099 of LNCS, pages 278–297. 2004.

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

[Lom08] Irina Lomazova. Nested petri nets for adaptive process modeling. In A. Avron, N. Dershowitz, and A. Rabinovich, editors, *Pillars of Computer Science*, volume 4800 of LNCS, pages 460–474. Springer, 2008.

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

[Val91] Rüdiger Valk. Modelling concurrency by task/flow EN systems. In *3rd Workshop on Concurrency and Compositionality*, number 191 in GMD-Studien, St. Augustin, Bonn, 1991. Gesellschaft für Mathematik und Datenverarbeitung.

[Val03] Rüdiger Valk. Object Petri nets: Using the nets-within-nets paradigm. In Jörg Desel, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advanced Course on Petri Nets 2003*, volume 3098 of LNCS, pages 819–848. 2003.