Counting Abstraction and Decidability for the Verification of Structured Parameterized Networks

Marius Bozga[1], Radu Iosif[1], Arnaud Sangnier[2], and Neven Villani[1]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, VERIMAG, 38000, France
{firstname}.{lastname}@univ-grenoble-alpes.fr
[2] DIBRIS, Univ. of Genova, Italy
{firstname}.{lastname}@unige.it

Abstract. We consider the verification of parameterized networks of replicated processes whose architecture is described by hyperedge-replacement graph grammars in the style of Courcelle. Due to the undecidability of verification problems such as reachability or coverability of a given configuration, in which we count the number of replicas in each local state, we develop two orthogonal verification techniques. We present a counting abstraction able to produce, from a graph grammar describing a parameterized system, a finite set of Petri nets that over-approximate the behaviors of the original system. The counting abstraction is implemented in a prototype tool, evaluated on a non-trivial set of test cases. Moreover, we identify a decidable fragment, for which the coverability problem is in 2EXPTIME and PSPACE-hard.

1 Introduction

The architecture is a crucial design aspect for the functionality of a computer network. For instance, the code of a consensus protocol changes depending on whether it is used in a ring or a clique-shaped network. The architecture also influences the traffic balance and overall efficiency of communication. Formal modelling of network architectures is a key enabler for the use of verification algorithms that prove absence of error scenarios in a distributed environment (*e.g.,* deadlocks, races or mutual exclusion violations), or convergence towards a desired goal (*e.g.,* building a spanning-tree or electing a leader).

The impressive size of present day networks requires *parameterized models*, that describe infinite families of networks having an unbounded number of nodes. The problem of *parameterized verification* (*i.e.,* proving correctness for any number of processes) often amounts to model-checking a small cut-off of the system (see [8] for a survey). In cases where a cut-off does not exist or is too large, symbolic representations of invariants (*i.e.,* sets of configurations closed under local and communication actions) using *e.g.,* boolean constraints [19], well-structured transitions systems [1], monadic second-order logic [11] or finite-state automata [28] can be used to decide a parameterized safety problem in a matter of seconds. This is because, in particular, parameterized verification methods do not suffer from the state explosion problem of classical model-checking techniques, that scale poorly in the number of concurrent processes.

However, a current limitation is that most techniques rely on hard-coded network topologies, typically cliques [22], rings [14] or combinations thereof [5]. Many archi-

tecture description languages have been developped by the software engineering community (see, *e.g.,* [13] and [15] for surveys) to support network design but, in general, these languages lack support for verification. Only few recent parameterized verification techniques take architectures as input of the problem, described using, *e.g.,* first-order [9] or separation logic [12]. Such descriptive (logic-based) languages are typically hard to use, because of the generality of their semantics, that requires complex frame conditions to specify what is actually *not* part of the architecture.

Contributions We consider the parametric verification problem for process networks specified by *graph grammars* that use operations of the standard *hyperedge-replacement* (HR) algebra of graphs [17] to describe how graphs are inductively build from smaller subgraphs. This constructive aspect of graph grammars makes them appealing for network design, because recursive specification of types and datastructures are widespread among programmers. Graph grammars are, moreover, at the core of a solid theory (see [17] for a comprehensive survey). In principle, HR graph grammars can specify families of graphs having bounded tree-width, such as chains, rings, stars, trees (of unbounded rank) and beyond, *e.g.,* overlaid structures such as trees or stars with certain nodes linked in a list. Since cliques and grids are families of unbounded tree-width, neither can be specified using HR graph grammars. However, the relation between the expressiveness of *vertex-replacement* (VR) and that of HR grammars[3] [**?**] could provide a generalization of our techniques to cliques, bipartite graphs and, in general, architectures specified using VR grammars. We consider this for future work.

Because the parameterized verification problem is undecidable, even for chain-like networks, we consider two orthogonal lines of work. First, following the seminal work of German and Sistla [22], where identities of processes communicating through rendezvous in a clique is ignored to keep only the number of processes in each state, we define a *counting abstraction* that folds the infinite set of networks specified using a HR grammar into a finite set of Petri nets, which subsumes the behaviors of the original set of networks. As a consequence, if a set of places is not covered by an execution of some of the resulting Petri nets, then it is not covered by the original set of behaviors. These coverability problems can be used to express mutual exclusion, and can encode other properties (*e.g.,* finite-valued consensus). Even though, computing the counting abstraction for clique networks is fairly simple [22], it is less trivial for families of networks specified by a grammar. We circumvent these technical challenges by defining appropriate HR algebras, in which the abstraction can be computed by a finite Kleene iteration of the grammar. This line of work is motivated by several recent advances on the theory [20] and tool support [31] for the reachability and coverability problem for Petri nets. The abstraction has been implemented in a prototype tool and a number of experiments showing the effectiveness of the method have been carried out.

Second, we define a decidable fragment of the original problem, by restricting the local behavior of the nodes to *pebble-passing systems*, where a finite (but unbounded) number of identical pebbles can be moved from one node to another. We inspire ourselves from token-passing systems[4,6] for which a restriction on the behavior of each proccess allows to get decidability results of some verification problems. Note that in

[3] Each VR grammar can be transformed into an HR grammar modulo an elimination of epsilon edges, similar to the epsilon edge elimination for finite automata.

our case, the processes definition are simple, but we allow an unbounded number of tokens/pebbles. Interestingly, our decidable restriction applies only to the local behavior and does not restrict the family of networks considered, other than that they must be the language of a given HR grammar. Examples of problems from this decidable fragment include token-rings, tree-traversal used, in general, for notification and binary consensus among the participants of general (HR-specified) architectures. We have studied the complexity of the decidable fragment and found it to be doubly-exponential in the unary size of the coverability property and in the maximal tree-width the set of networks generated by the grammar. At the same time, we found the problem to be PSPACE-hard.

Related Work Traditionally, verification of unbounded networks of parallel processes considers known architectural patterns, typically cliques or rings [22,14]. Because the price for decidability is drastic restriction on architecture styles [8], more recent works propose practical semi-algorithms, *e.g., regular model checking* [24] or *automata learning* [16]. Here the architecture is implicitly determined by the class of language recognizers: word automata encode pipelines or rings, whereas tree automata describe trees.

Specifying parameterized concurrent systems inductively is reminiscent of *network grammars* [29,26,23], that use inductive rules to describe systems with linear (pipeline, token-ring) architectures obtained by composition of an unbounded number of processes. In contrast, our language is based on the HR graph algebra. Verification of network grammars against safety properties (reachability of error configurations) requires the synthesis of *network invariants* [33], computed by costly fixpoint iterations [27] or by abstracting (forgetting the particular values of indices in) the composition of a small number of instances [25]. A more recent line of work considers a lightweight invariant synthesis method based on the inference of *structural invariants*[4] of an infinite family of Petri nets, for parameterized systems whose network architectures are specified using logic [9,12]. This method is geared towards deadlock-freedom and mutual exclusion, whereas our counting abstraction method works for coverability properties, in general.

Other methods to verify safety properties of parameterized networks consist in changing the semantics of behaviors to obtain an over-approximation having a decidable safety verification problem. In [2], the authors have implemented such a scheme using a *monotonic abstraction*, where the resulting abstraction is a *well-structured transition systems* [1]. They first consider pipeline architectures, where communication is done by checking existentially or universally the state of the other proceess. In [3], a similar technique is applied to networks with clique architectures, where processes can manipulate shared boolean and natural variables. In contrast, both our methods target network architectures defined by unrestricted HR graph grammars.

2 Preliminaries

We denote by $\mathbb{N}$ the set of positive integers, including zero. For $i, j \in \mathbb{N}$, we denote by $[i, j]$ the set $\{i, \ldots, j\}$, considered empty if $i > j$. The cardinality of a finite set A is denoted $\|A\|$. A singleton $\{a\}$ will be denoted a. By $A \subseteq_{fin} B$, we mean that A is a finite subset of B. The union of two disjoint sets A and B is denoted as $A \uplus B$. The Cartesian

[4] Invariants that depend on the structure of a Petri net, which hold for any of its executions.

product of two sets A and B is denoted $A \times B$. As usual, we denote by A^* the set of (possibly empty) sequences of elements from A.

For a function $f : A \to B$ and $C \subseteq A$, we write $f(C) \stackrel{\text{def}}{=} \{f(c) \mid c \in C\}$ and $f\downarrow_C \stackrel{\text{def}}{=} \{(c, f(c)) \mid c \in C\}$. The inverse of a function $f : A \to B$ is the relation $f^{-1} \stackrel{\text{def}}{=} \{(f(a), a) \mid a \in A\}$. To alleviate notation, we write $f(x,y)$ instead of $f((x,y))$, when no confusion arises. For two functions $f : A \to B$ and $g : C \to D$, the function $f \times g : A \times B \to C \times D$ maps each pair $(a,c) \in A \times C$ into the pair $(f(a), g(c)) \in B \times D$. A bijective function $f : A \to A$ is a *finite permutation* if the set $\{a \in A \mid f(a) \neq a\}$ is finite. In particular, $a \leftrightarrow b$ denotes the finite permutation that switches a with b and leaves the other elements of the domain unchanged. A finite partial function is denoted as $f : A \rightharpoonup_{fin} B$ and $\mathrm{dom}(f)$, $\mathrm{img}(f)$ denote its domain and range, respectively. When $f : A \to C$ and $g : B \to C$ coincide on their shared domain $A \cap B$, we write $f \cup g : A \cup B \to C$ the function such that $(f \cup g)\downarrow_A = f$ and $(f \cup g)\downarrow_B = g$. The full version of the paper contains proofs of technical results [10].

2.1 Petri Nets

A *net* is a tuple $\mathsf{N} = (\mathsf{Q}, \mathsf{T}, \mathsf{W})$, where Q is a finite set of *places*, T is a finite set of *transitions* such that $\mathsf{Q} \cap \mathsf{T} = \emptyset$ and $\mathsf{W} : (\mathsf{Q} \times \mathsf{T}) \cup (\mathsf{T} \times \mathsf{Q}) \to \mathbb{N}$ is a *weighted incidence relation* between places and transitions. We denote by Q_N, T_N and W_N the places, transitions and incidence relation of N, respectively. For all $x, y \in \mathsf{Q} \cup \mathsf{T}$ such that $\mathsf{W}(x,y) > 0$, we say that there is an *edge of weight* $\mathsf{W}(x,y)$ between x and y. For an element $x \in \mathsf{Q} \cup \mathsf{T}$, we define the set of *predecessors* ${}^\bullet x \stackrel{\text{def}}{=} \{y \in \mathsf{Q} \cup \mathsf{T} \mid \mathsf{W}(y,x) > 0\}$, *successors* $x^\bullet \stackrel{\text{def}}{=} \{y \in \mathsf{Q} \cup \mathsf{T} \mid \mathsf{W}(x,y) > 0\}$ and predecessor-successor pair ${}^\bullet x^\bullet \stackrel{\text{def}}{=} ({}^\bullet x, x^\bullet)$.

A *marking* of $\mathsf{N} = (\mathsf{Q}, \mathsf{T}, \mathsf{W})$ is a function $\mathsf{m} : \mathsf{Q} \to \mathbb{N}$. A transition t is *enabled* in m if $\mathsf{m}(q) \geq \mathsf{W}(q,t)$, for each place $q \in \mathsf{Q}$. For all markings m, m' and transitions $t \in \mathsf{T}$, we write $\mathsf{m} \stackrel{t}{\rightsquigarrow} \mathsf{m}'$ when t is enabled in m and $\mathsf{m}'(q) = \mathsf{m}(q) - \mathsf{W}(q,t) + \mathsf{W}(t,q)$, for all $q \in \mathsf{Q}$. Given two markings m and m', a finite sequence of transitions $\mathbf{t} = (t_1, \ldots, t_n)$ is a *firing sequence*, written $\mathsf{m} \stackrel{\mathbf{t}}{\rightsquigarrow} \mathsf{m}'$, if and only if either (i) $n = 0$ and $\mathsf{m} = \mathsf{m}'$, or (ii) $n \geq 1$ and there exist markings $\mathsf{m}_1, \ldots, \mathsf{m}_{n-1}$ such that $\mathsf{m} \stackrel{t_1}{\rightsquigarrow} \mathsf{m}_1 \stackrel{t_2}{\rightsquigarrow} \ldots \stackrel{t_{n-1}}{\rightsquigarrow} \mathsf{m}_{n-1} \stackrel{t_n}{\rightsquigarrow} \mathsf{m}'$. A sequence $\mathbf{t}$ is *fireable* from m whenever there exists a marking m' such that $\mathsf{m} \stackrel{\mathbf{t}}{\rightsquigarrow} \mathsf{m}'$.

A *Petri net* (PN) is a pair $\mathcal{N} = (\mathsf{N}, \mathsf{m}_0)$, where N is a net and m_0 is the *initial marking* of N. For simplicity, we write $\mathsf{Q}_\mathcal{N} \stackrel{\text{def}}{=} \mathsf{Q}_\mathsf{N}$, $\mathsf{T}_\mathcal{N} \stackrel{\text{def}}{=} \mathsf{T}_\mathsf{N}$, $\mathsf{W}_\mathcal{N} \stackrel{\text{def}}{=} \mathsf{W}_\mathsf{N}$ and $\mathrm{init}_\mathcal{N} \stackrel{\text{def}}{=} \mathsf{m}_0$ for the elements of $\mathcal{N}$. A marking m is *reachable* in $\mathcal{N}$ iff there exists a firing sequence $\mathbf{t}$ such that $\mathsf{m}_0 \stackrel{\mathbf{t}}{\rightsquigarrow} \mathsf{m}$. We denote by $\mathrm{reach}(\mathcal{N})$ the set of reachable markings of $\mathcal{N}$. The *reachability problem* asks, given a PN $\mathcal{N}$ and a marking m, does $\mathsf{m} \in \mathrm{reach}(\mathcal{N})$? The *coverability problem* asks, for a given PN $\mathcal{N}$ and marking m, does there exists a marking $\mathsf{m}' \in \mathrm{reach}(\mathcal{N})$ such that $\mathsf{m} \leq \mathsf{m}'$? Here the order of markings is the pointwise order on $\mathbb{N}$, *i.e.,* $\mathsf{m} \leq \mathsf{m}'$ iff $\mathsf{m}(q) \leq \mathsf{m}'(q)$ for all $q \in \mathsf{Q}_\mathcal{N}$. The coverability problem is more concisely stated using the set of covered markings $\mathrm{cover}(\mathcal{N}) \stackrel{\text{def}}{=} \{\mathsf{m} \mid \exists \mathsf{m}' \in \mathrm{reach}(\mathcal{N}) \,.\, \mathsf{m} \leq \mathsf{m}'\}$, *i.e.,* given $\mathcal{N}$ and m, does $\mathsf{m} \in \mathrm{cover}(\mathcal{N})$?

2.2 Parameterized Systems

We begin by defining parameterized communicating systems, *i.e.,* graphs whose vertices model network nodes that run identical copies of one or more process types. Neighbouring processes synchronize their transitions according to the observable edge labels of the network graph. In most of the literature (see, *e.g.,* [7] for a survey) process types are represented by finite labeled transition systems (LTS), *e.g.,* with disjoint observable and internal alphabets of transition labels. For simplicity, here we use PNs whose transitions mimick closely the transitions of a LTS, thus avoiding the formal definition of the latter.

Let Λ and Δ be finite disjoint alphabets of vertex and edge labels, respectively. A (binary labeled) *graph* is a tuple $\mathsf{G} = (\mathsf{V}, \mathsf{E}, \lambda)$, where V is a finite set of vertices, $\mathsf{E} \subseteq \mathsf{V} \times \Delta \times \mathsf{V}$ is a set of labeled binary edges and $\lambda : \mathsf{V} \to \Lambda$ maps each vertex to a vertex label. Edges (v_1, a, v_2) are written $v_1 \xrightarrow{a} v_2$. We denote by $\mathsf{V_G}$, $\mathsf{E_G}$ and λ_G the vertices, edges and vertex labeling of G, respectively. We do not distinguish isomorphic graphs, *i.e.,* graphs that differ only in the identities of their vertices.

Definition 1. *A* process type p *is a PN having weights at most* 1 *and exactly one marked place initially, whose transitions are partitioned into* observable $\mathsf{T}_\mathsf{p}^{obs}$ *and* internal $\mathsf{T}_\mathsf{p}^{int}$*, i.e.,* $\mathsf{T_p} = \mathsf{T}_\mathsf{p}^{obs} \uplus \mathsf{T}_\mathsf{p}^{int}$*, and each transition has exactly one predecessor and one successor. Let* $\mathcal{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_k\}$ *be a finite fixed set of process types such that* $\mathsf{Q}_{\mathsf{p}_i} \neq \emptyset$*, for all* $i \in [1,k]$ *and* $\mathsf{Q}_{\mathsf{p}_i} \cap \mathsf{Q}_{\mathsf{p}_j} = \emptyset$*, for all* $1 \leq i < j \leq k$*.*

Because a process type has exactly one initial token and all transitions have one predecessor and one successor, both with weight exactly 1, every reachable marking of a process type has exactly one initial token. A PN having this property is said to be *automata-like*. We denote by $\mathsf{Q}_\mathcal{P}$ and $\mathsf{T}_\mathcal{P}^{obs}$ the sets of places and observable transitions from some $\mathsf{p} \in \mathcal{P}$, respectively.

Example 1. Figure 1 (a) shows two process types *Cont* and *Proc*. They both represent entities that can hold a token and they can either grab a token, if they do not have it, or release the token, otherwise. The first one, which we identify as a controller, has only observable transitions, whereas the second one, which represents a worker process, has two internal trasitions *start* and *stop* depicted in yellow. These transitions are used to simulate the fact that when the worker has the token, it can move to a working state, from which he cannot release the token and when it stops working, it can move back to a state from which the token can be released.

Definition 2. *A* system $\mathsf{S} = (\mathsf{V}, \mathsf{E}, \lambda)$ *is a graph whose vertices are labeled with process types from* $\mathcal{P}$ *(*$\Lambda = \mathcal{P}$*) and edges with pairs of observable transitions from* $\mathsf{T}_\mathcal{P}^{obs}$ *(i.e.,* $\Delta = \mathsf{T}_\mathcal{P}^{obs} \times \mathsf{T}_\mathcal{P}^{obs}$*), such that* $t_i \in \mathsf{T}_{\lambda(v_i)}^{obs}$*, for both* $i = 1,2$*, for each edge* $v_1 \xrightarrow{(t_1,t_2)} v_2 \in \mathsf{E_S}$*.*

Example 2. Figure 1 (b) shows two systems labeled with the process types given by Figure 1 (a). They both represent a network with four entities, one controller of type *Cont* and three working processes of type *Proc*. In S_1, the controller can pass a token to the first working process, which can pass the token to the second one which can pass the token to the third one. In S_2, the controller is at the center and it communicate with all the working processes that get and release the token.

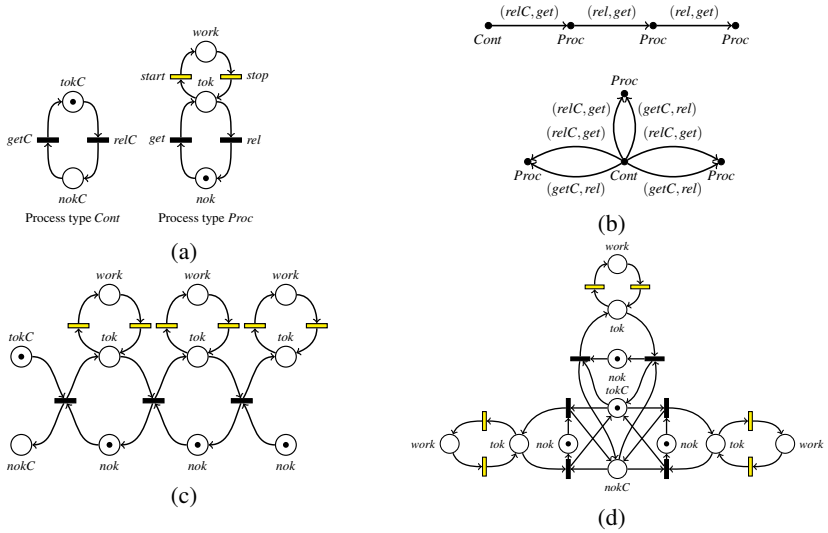


Fig. 1. Two process types (a), A system with chain-shape network S_1 (top) and a system with star-shape network S_2 (bottom) (b). Behavior of S_1 (c). Behavior of S_2 (d)

The communication (*i.e.,* synchronization between processes) in a system is formally captured by the following notion of behavior:

Definition 3. *A* behavior *is a PN* $\mathcal{N}$ *such that* $1 \leq \|{}^\bullet t\| = \|t^\bullet\| \leq 2$*, for each* $t \in \mathsf{T}_\mathcal{N}$*. The* behavior of a system $\mathsf{S} = (\mathsf{V}, \mathsf{E}, \lambda)$ *is* $\beta(\mathsf{S}) \stackrel{\text{def}}{=} (\mathsf{N}, \mathsf{m}_0)$*, where:*

- $\mathsf{Q}_\mathsf{N} \stackrel{\text{def}}{=} \{(q,v) \mid q \in \mathsf{Q}_{\lambda(v)},\ v \in \mathsf{V}\}$*, a place* (q,v) *corresponds to the place* q *of the process type* $\lambda(v)$ *that labels the vertex* v*;*
- $\mathsf{T}_\mathsf{N} \stackrel{\text{def}}{=} \mathsf{E} \cup \{(t,v) \mid t \in \mathsf{T}^{int}_{\lambda(v)},\ v \in \mathsf{V}\}$*, the transitions are either edges of the system (i.e., modeling the synchronizations of two processes) or pairs* (t,v) *corresponding to an internal transition* t *of the process type* $\lambda(v)$ *that labels the vertex* v*;*
- *the weight function* W_N *is defined below:*

$$\mathsf{W}_\mathsf{N}((q,v), v_1 \xrightarrow{(t_1,t_2)} v_2) \stackrel{\text{def}}{=} \begin{cases} \mathsf{W}_{\lambda(v)}(q,t_i)\text{, if } v = v_i \text{, for } i = 1,2 \\ 0\text{, otherwise} \end{cases}$$

$$\mathsf{W}_\mathsf{N}(v_1 \xrightarrow{(t_1,t_2)} v_2, (q,v)) \stackrel{\text{def}}{=} \begin{cases} \mathsf{W}_{\lambda(v)}(t_i,q)\text{, if } v = v_i \text{, for } i = 1,2 \\ 0\text{, otherwise} \end{cases}$$

$$\mathsf{W}_\mathsf{N}((q,v),(t,v')) \stackrel{\text{def}}{=} \begin{cases} \mathsf{W}_{\lambda(v)}(q,t)\text{, if } v = v' \\ 0\text{, otherwise} \end{cases} \quad \mathsf{W}_\mathsf{N}((t,v'),(q,v)) \stackrel{\text{def}}{=} \begin{cases} \mathsf{W}_{\lambda(v)}(t,q)\text{, if } v = v' \\ 0\text{, otherwise} \end{cases}$$

- $\mathsf{m}_0(q,v) \stackrel{\text{def}}{=} \text{init}_{\lambda(v)}(q)$*, for all* $v \in \mathsf{V}$ *and* $q \in \mathsf{Q}_{\lambda(v)}$*.*

For example, Figure 1 (c) and (d) show the behaviors of the systems from Figure 1 (b). By construction, among places $\{(q,v) \mid q \in \mathsf{Q}_{\lambda(v)}\}$ for any v, there is exactly one token

in all reachable markings because $\lambda(v)$ is automata-like. By extension we say that such behaviors are automata-like.

A *parameterized system* $\mathbf{S} = \{\mathsf{S}_1, \mathsf{S}_2, \ldots\}$ is a possibly infinite set of systems, called *instances*. A parameterized system has an infinite set of behaviors, denoted as $\beta(\mathbf{S})$, *i.e.,* one for each instance. In the rest of this paper we are concerned with the following parameterized verification problems:

Definition 4. *The* parameterized reachability (*resp.* coverability) problem *takes in input a parameterized system* $\mathbf{S}$*, a set of places* $Q \subseteq \mathsf{Q}_{\mathcal{P}}$ *and a mapping* $\mathsf{m} : Q \to \mathbb{N}$ *and asks for the existence of an instance* $\mathsf{S} \in \mathbf{S}$ *and of a marking* $\overline{\mathsf{m}} \in \mathrm{reach}(\beta(\mathsf{S}))$ *(resp.* $\overline{\mathsf{m}} \in \mathrm{cover}(\beta(\mathsf{S}))$*) where* $\sum_{\{v \in \mathsf{V}_\mathsf{S} \mid q \in \mathsf{Q}_{\lambda_\mathsf{S}(v)}\}} \overline{\mathsf{m}}(q,v) = \mathsf{m}(q)$*, for all* $q \in Q$*.*

For Petri nets, the reachability problem specified by the number of token within a set of places is known as the *submarking reachability* [**?**]. For example, a typical correctness property that can be stated as a parameterized coverability problem is mutual exclusion: a parameterized system violates mutual exclusion if there is an instance that reaches a marking in which two or more processes of the same type are in the same given local place.

Example 3. For the two systems depicted on Figure 1, no two processes (*i.e.,* running on different nodes) work at the same time if and only if the parameterized coverability problem for $\mathsf{Q} = \{work\}$ and $\mathsf{m} = \{(work, 2)\}$ has a negative answer, *i.e.,* some marking that agrees with m over *work* cannot be covered by any marking reachable in the behavior of some instance.

2.3 Algebras

We recall a few notions on algebras needed in the following. A *signature* is a set of function symbols $\mathsf{F} = \{\mathsf{op}_1, \mathsf{op}_2, \ldots\}$. An F-*term* is a term built with function symbols from F and variables of arity zero. An F-term is *ground* if it has no variables. An F-*algebra* $\mathcal{A} = (\mathbb{A}, \mathsf{op}_1^{\mathcal{A}}, \mathsf{op}_2^{\mathcal{A}}, \ldots)$ interprets the function symbols from F as functions over the *domain* $\mathbb{A}$. Given F-algebras $\mathcal{A}$ and $\mathcal{B}$ having domains $\mathbb{A}$ and $\mathbb{B}$, respectively, a *homomorphism* is a function $h : \mathbb{A} \to \mathbb{B}$ such that $h(\mathsf{op}^{\mathcal{A}}(a_1, \ldots, a_n)) = \mathsf{op}^{\mathcal{B}}(h(a_1), \ldots, h(a_n))$, for each function symbol $\mathsf{op} \in \mathsf{F}$ of arity n and $a_1, \ldots, a_n \in \mathbb{A}$. The *kernel* of a function $f : \mathbb{A} \to \mathbb{B}$ is the equivalence relation $\sim_{\ker(f)} \subseteq \mathbb{A} \times \mathbb{A}$ defined as $a_1 \sim_{\ker(f)} a_2 \iff f(a_1) = f(a_2)$. An equivalence relation $\sim \subseteq \mathbb{A} \times \mathbb{A}$ is an F-*congruence* if and only if, for each function symbol $\mathsf{op} \in \mathsf{F}$ of arity n and $a_1, a_1', \ldots, a_n, a_n' \in \mathbb{A}$ such that $a_i \sim a_i'$, for all $i \in [1,n]$, we have $\mathsf{op}^{\mathcal{A}}(a_1, \ldots, a_n) \sim \mathsf{op}^{\mathcal{A}}(a_1', \ldots, a_n')$.

Proposition 1. *Let* $\mathcal{A}$ *be an* F*-algebra having domain* $\mathbb{A}$*, and* $f : \mathbb{A} \to \mathbb{B}$ *be a function such that* $\sim_{\ker(f)}$ *is an* F*-congruence. Then* f *is a homomorphism between* $\mathcal{A}$ *and* $\mathcal{B} \stackrel{\mathsf{def}}{=} (f(\mathbb{A}), \{\mathsf{op}^{\mathcal{B}}\}_{\mathsf{op} \in \mathsf{F}})$*, where* $\mathsf{op}^{\mathcal{B}}(b_1, \ldots, b_n) \stackrel{\mathsf{def}}{=} f(\mathsf{op}^{\mathcal{A}}(f^{-1}(b_1), \ldots, f^{-1}(b_n)))$[5]*, for each function symbol* $\mathsf{op} \in \mathsf{F}$ *of arity* n *and all* $b_1, \ldots, b_n \in f(\mathbb{A})$*. Consequently,* $f(\theta^{\mathcal{A}}) = \theta^{\mathcal{B}}$*, for each ground* F*-term* θ*.*

[5] The right-hand side of this definition is a singleton that we identify with its element.

Proof. First, we prove that $\mathsf{op}^{\mathcal{B}}(b_1,\ldots,b_n)$ is well-defined and evaluates to a singleton, for each $\mathsf{op} \in \mathsf{F}$ and $b_1,\ldots,b_n \in f(\mathbb{A})$. Let $a_i \in f^{-1}(b_i)$ be elements, for all $i \in [1,n]$. The element a_i exists, by the choice of $b_i \in f(\mathbb{A})$, for all $i \in [1,n]$. Then $f^{-1}(b_i)$ is the $\sim_{\ker(f)}$-equivalence class of a_i, for all $i \in [1,n]$. Because $\sim_{\ker(f)}$ is an F-congruence, $\mathsf{op}^{\mathcal{A}}(f^{-1}(b_1),\ldots,f^{-1}(b_n))$ is the $\sim_{\ker(f)}$-equivalence class of $\mathsf{op}^{\mathcal{A}}(a_1,\ldots,a_n)$, hence $f(\mathsf{op}^{\mathcal{A}}(f^{-1}(b_1),\ldots,f^{-1}(b_n))) = \{f(\mathsf{op}^{\mathcal{A}}(a_1),\ldots,\mathsf{op}^{\mathcal{A}}(a_n))\}$ is a singleton. Second, to prove that f is an homomorphism between $\mathcal{A}$ and $\mathcal{B}$, we compute, for each $\mathsf{op} \in \mathsf{F}$ and all $a_1,\ldots,a_n \in \mathbb{A}$:

$$\begin{aligned}\mathsf{op}^{\mathcal{B}}(f(a_1),\ldots,f(a_n)) &= f(\mathsf{op}^{\mathcal{A}}(f^{-1}(f(a_1)),\ldots,f^{-1}(f(a_n)))), \text{ by the definition of } \mathsf{op}^{\mathcal{B}}\\ &= f(\mathsf{op}^{\mathcal{A}}(a_1,\ldots,a_n)), \text{ because } \sim_{\ker(f)} \text{ is an } \mathsf{F}\text{-congruence}\end{aligned}$$

Finally, $f(\theta^{\mathcal{A}}) = \theta^{\mathcal{B}}$ holds for each ground F-term θ, because f is a homomorphism between $\mathcal{A}$ and $\mathcal{B}$. □

We introduce a standard signature of operations on graphs and define two algebras, of open systems and behaviors. Let Σ be a countably infinite set of *source labels*, fixed in the rest of the paper. With no loss of generality, we assume that Σ is partitioned into disjoint sets indexed by the process types $\mathcal{P} = \{\mathsf{p}_1,\ldots,\mathsf{p}_k\}$, *i.e.,* $\Sigma = \Sigma_{\mathsf{p}_1} \uplus \ldots \uplus \Sigma_{\mathsf{p}_k}$. A source label $\sigma \in \Sigma$ uniquely identifies a process type $\mathrm{ptype}(\sigma) \in \mathcal{P}$ such that $\sigma \in \Sigma_{\mathrm{ptype}(\sigma)}$. A function $\alpha : \Sigma \to \Sigma$ is *$\mathcal{P}$-preserving* if $\mathrm{ptype}(\alpha(\sigma)) = \mathrm{ptype}(\sigma)$, for all $\sigma \in \Sigma$.

The signature of the *hyperedge-replacement* graph algebra (HR) [17] consists of the constants a_{σ_1,σ_2}, for all edge labels $a \in \Delta$ and source labels $\sigma_1,\sigma_2 \in \Sigma$, the unary symbols $\mathsf{restrict}_\tau$, for all $\tau \subseteq_{fin} \Sigma$, rename_α, for all $\mathcal{P}$-preserving finite permutations $\alpha : \Sigma \to \Sigma$ and the binary symbol $\oplus$. By HR congruence we mean an equivalence relation that is a congruence for the HR signature.

An *open system* is a pair $\check{\mathsf{S}} \stackrel{\mathsf{def}}{=} (\mathsf{S},\xi)$, where $\mathsf{S} = (\mathsf{V},\mathsf{E},\lambda)$ is a system and $\xi : \Sigma \rightharpoonup_{fin} \mathsf{V}_{\mathsf{S}}$ is an injective partial function that assigns source labels to vertices of S such that $\mathrm{ptype}(\sigma) = \lambda(\xi(\sigma))$, for each $\sigma \in \mathrm{dom}(\xi)$, *i.e.,* the source label of a vertex has the process type of that vertex. We say that a vertex $v \in \mathsf{V}_{\mathsf{S}}$ is the σ-source of $\check{\mathsf{S}}$ if $\xi(\sigma) = v$. The *type* of $\check{\mathsf{S}}$ is $\mathrm{type}(\check{\mathsf{S}}) \stackrel{\mathsf{def}}{=} \mathrm{dom}(\xi)$, *i.e.,* the set of source labels that occurs in S. Since systems (Definition 2) are in fact open systems of empty type, we blur the distinction and refer to open systems as systems from now on.

The algebra $\mathcal{S}$ (Figure 2) interprets the HR signature over the set $\mathbb{S}$ of systems:

- $a^{\mathcal{S}}_{\sigma_1,\sigma_2}$ is the system having a single a-labeled edge between its two vertices labeled with process types $\mathrm{ptype}(\sigma_1)$ and $\mathrm{ptype}(\sigma_2)$, that are the σ_1- and σ_2-sources, respectively, for each edge label $a \in \Delta$ and source labels $\sigma_1,\sigma_2 \in \Sigma$.
- $\mathsf{restrict}^{\mathcal{S}}_\tau(\mathsf{S},\xi) \stackrel{\mathsf{def}}{=} (\mathsf{S},\xi\downarrow_\tau)$ removes the source labels that are not in $\tau \subseteq_{fin} \Sigma$,
- $\mathsf{rename}^{\mathcal{S}}_\alpha(\mathsf{S},\xi) \stackrel{\mathsf{def}}{=} (\mathsf{S},\xi \circ \alpha^{-1})$ relabels the sources according to α; note that $\xi \circ \alpha^{-1}$ is an injective partial mapping, since α is a finite permutation of Σ,
- $(\mathsf{S}_1,\xi_1) \oplus^{\mathcal{S}} (\mathsf{S}_2,\xi_2)$ is the disjoint union of (S_1,ξ_1) and (S_2,ξ_2), followed by:
  * fusion of each pair of common σ-sources $v_i \in \mathsf{V}_{\mathsf{S}_i}$, for $\sigma \in \mathrm{type}(\mathsf{S}_1) \cap \mathrm{type}(\mathsf{S}_2)$ and $i = 1,2$, into a σ-source labeled with $\mathrm{ptype}(\sigma)$,
  * fusion of all identical edges into a single edge with the same label and endpoints.

Every other HR algebra considered in the rest of this paper will be defined from $\mathcal{S}$ via an homomorphism. Figure 2 shows the diagram of these algebras and homomorphisms.

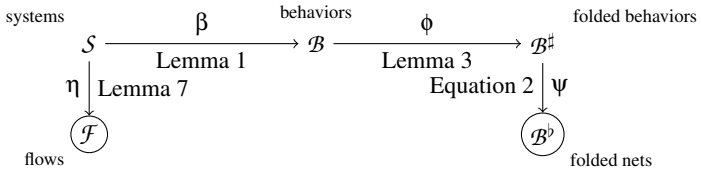


Fig. 2. Homomorphisms between the HR algebras used in this paper. The circled algebras are the finite and effectively computable ones.

Each such homomorphism h (except for ψ, which has a trivial definition) has a reference to a lemma proving that $\sim_{\ker(h)}$ is an HR congruence.

An *open behavior* is a pair $\check{\mathcal{N}} \stackrel{\text{def}}{=} (\mathcal{N}, \xi)$, where $\mathcal{N}$ is a behavior and $\xi : \Sigma \times \mathsf{Q}_{\mathcal{P}} \rightharpoonup_{fin} \mathsf{Q}_{\mathcal{N}}$ is an injective partial function assigning pairs (σ, q) to places of $\mathcal{N}$, where σ is a source label and q is a place from some process type in $\mathcal{P}$. A place $r \in \mathsf{Q}_{\mathcal{N}}$ is a (σ, q)-source of $\check{\mathcal{N}}$ if $\xi(\sigma, q) = r$ and $\text{type}(\check{\mathcal{N}}) \stackrel{\text{def}}{=} \text{dom}(\xi)$ denotes the type of $\check{\mathcal{N}}$. Since behaviors (Definition 3) are actually open behaviors of empty type, we blur the distinction and refer to open behaviors as behaviors, when no confusion arises.

To define the algebra of behaviors, we extend the β function introduced by Definition 3 to (open) systems, as follows. The behavior of the system $\check{\mathsf{S}} = (\mathsf{S}, \xi)$ is $\beta(\check{\mathsf{S}}) \stackrel{\text{def}}{=} (\beta(\mathsf{S}), \overline{\xi})$, where $\beta(\mathsf{S})$ is given in Definition 3 and the source labeling $\overline{\xi}$ is $\overline{\xi}(\sigma, q) \stackrel{\text{def}}{=} (q, \xi(\sigma))$ if $\xi(\sigma)$ is defined and $(q, \xi(\sigma)) \in \mathsf{Q}_{\mathcal{N}}$, and undefined otherwise, for all $\sigma \in \Sigma$ and $q \in \mathsf{Q}_{\mathcal{P}}$.

Lemma 1. *$\sim_{\ker(\beta)}$ is a HR congruence.*

Proof. We prove a stronger statement, namely that β is injective. Then, $\sim_{\ker(\beta)}$ becomes the equality, which is trivially a congruence for any signature. Let $\check{\mathsf{S}}_i = (\mathsf{S}_i, \xi_i)$ be open systems and $\beta(\check{\mathsf{S}}_i) = ((\mathsf{N}_i, \mathsf{m}_{0i}), \overline{\xi}_i)$ be open behaviors, for $i = 1, 2$, such that $\mathsf{N}_1 = \mathsf{N}_2$, $\mathsf{m}_{01} = \mathsf{m}_{02}$ and $\overline{\xi}_1 = \overline{\xi}_2$. We show that $\mathsf{S}_1 = \mathsf{S}_2$ below:

- $\mathsf{V}_{\mathsf{S}_1} = \mathsf{V}_{\mathsf{S}_2}$: Suppose, for a contradiction, that there exists a vertex $v \in \mathsf{V}_{\mathsf{S}_1} \setminus \mathsf{V}_{\mathsf{S}_2}$. Since $\mathsf{Q}_{\lambda_{\mathsf{S}_1}(v)} \neq \emptyset$ (Definition 1), there exists a place $q \in \mathsf{Q}_{\lambda_{\mathsf{S}_1}(v)}$. Since $\mathsf{Q}_{\mathsf{N}_1} = \mathsf{Q}_{\mathsf{N}_2}$, by Definition 3, we obtain $q \in \mathsf{Q}_{\lambda_{\mathsf{S}_2}}(v)$ and $v \in \mathsf{V}_{\mathsf{S}_2}$, contradiction. Hence $\mathsf{V}_{\mathsf{S}_1} \subseteq \mathsf{V}_{\mathsf{S}_2}$ and the proof in the other direction is symmetric.
- $\lambda_{\mathsf{S}_1} = \lambda_{\mathsf{S}_2}$: Let $v \in \mathsf{V}_{\mathsf{S}_1}$ ($= \mathsf{V}_{\mathsf{S}_2}$, by the previous point) and let $q \in \mathsf{Q}_{\lambda_{\mathsf{S}_1}(v)}$ be a place. Since $\mathsf{Q}_{\mathsf{N}_1} = \mathsf{Q}_{\mathsf{N}_2}$, by definition Definition 3, we obtain $q \in \mathsf{Q}_{\lambda_{\mathsf{S}_2}(v)}$, hence $\lambda_{\mathsf{S}_1}(v) = \lambda_{\mathsf{S}_2}(v)$, by Definition 1, *i.e.,* because the process types are assumed to have pairwise disjoint sets of places. Since the choice of v was arbitrary, we obtain $\lambda_{\mathsf{S}_1} = \lambda_{\mathsf{S}_2}$.
- $\mathsf{E}_{\mathsf{S}_1} = \mathsf{E}_{\mathsf{S}_2}$: since $\mathsf{T}_{\mathsf{N}_1} = \mathsf{T}_{\mathsf{N}_2}$, $\mathsf{V}_{\mathsf{S}_1} = \mathsf{V}_{\mathsf{S}_2}$ and $\lambda_{\mathsf{S}_1} = \lambda_{\mathsf{S}_2}$, by the previous points, we obtain $\mathsf{E}_{\mathsf{S}_1} = \mathsf{E}_{\mathsf{S}_2}$, by Definition 3.

To prove $\xi_1 = \xi_2$, let $\sigma \in \Sigma$ be a source label and $q \in \mathsf{Q}_{\mathcal{P}}$ be a place from some process type, such that $\overline{\xi}_1(\sigma, q)$ is defined. Then, $\overline{\xi}_1(\sigma, q) = (q, \xi_1(\sigma))$. Since $\overline{\xi}_1 = \overline{\xi}_2$, we obtain that $\overline{\xi}_2(\sigma, q)$ is defined and $\xi_1(\sigma) = \xi_2(\sigma)$. Since the choice of σ was arbitrary, we obtain $\xi_1 = \xi_2$. □

We introduce an algebra $\mathcal{B}$ of behaviors (Figure 2), whose domain and interpretation of HR function symbols are defined as in Proposition 1. Since $\sim_{\ker(\beta)}$ is a HR congruence (Lemma 1), it follows that β is a homomorphism between $\mathcal{S}$ and $\mathcal{B}$. As we discuss next, a grammar that describes a parameterized system $\mathbf{S} = \{\mathsf{S}_1, \mathsf{S}_2, \ldots\}$ can be reused to describe its set of behaviors $\beta(\mathbf{S})$.

2.4 Grammars

A *grammar* over a signature F is a pair $\Gamma = (\Xi, \Pi)$ consisting of a finite set Ξ of *nonterminals* and a finite set Π of *rules* of the form, either (1) $X \to \rho[X_1, \ldots, X_n]$, where $X, X_1, \ldots, X_n \in \Xi$ are nonterminals and ρ is an F-term whose only variables are $X_1, \ldots, X_n$, or (2) $\to X$, where $X \in \Xi$; the rules of this form are called *axioms*. Given terms θ and η, a *step* $\theta \Rightarrow_\Gamma \eta$ obtains η from θ by replacing an occurrence of a nonterminal X with the term ρ, for some rule $X \to \rho$ of Γ. An *X-derivation* is a sequence of steps starting with a nonterminal X. The derivation is *complete* if it ends in a ground term. Let $\mathcal{L}^{\mathcal{A}}_X(\Gamma) \stackrel{\text{def}}{=} \{\theta^{\mathcal{A}} \mid X \Rightarrow^*_\Gamma \theta \text{ is a complete derivation}\}$ and $\mathcal{L}^{\mathcal{A}}(\Gamma) \stackrel{\text{def}}{=} \bigcup_{\to X \in \Pi} \mathcal{L}^{\mathcal{A}}_X(\Gamma)$ be the *language* of Γ in the algebra $\mathcal{A}$.

The following result, also known as the *Filtering Theorem*, allows to build a grammar for the intersection between the language of a grammar and a *recognizable set*, *i.e.,* the image of a finite set via an inverse homomorphism [17, Theorem 3.88]:

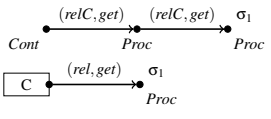
Theorem 1. *Let $\mathsf{F} = \{\mathsf{op}_1, \mathsf{op}_2, \ldots\}$ be a signature, $\mathcal{A} = (\mathbb{A}, \mathsf{op}_1^{\mathcal{A}}, \mathsf{op}_2^{\mathcal{A}}, \ldots)$ and $\mathcal{B} = (\mathbb{B}, \mathsf{op}_1^{\mathcal{B}}, \mathsf{op}_2^{\mathcal{B}}, \ldots)$ be F-algebras, such that $\mathbb{B}$ is finite, and h be a homomorphism between $\mathcal{A}$ and $\mathcal{B}$. For each F-grammar Γ and set $C \subseteq \mathbb{B}$, one can build a grammar $\Gamma_{h,C}$ such that $\mathcal{L}^{\mathcal{A}}(\Gamma_{h,C}) = \mathcal{L}^{\mathcal{A}}(\Gamma) \cap h^{-1}(C)$.*

The construction of the filtered grammar $\Gamma_{h,C}$ is effective: for each rule $X \to \rho[X_1, \ldots, X_n]$ of Γ and each sequence of elements $b, b_1, \ldots, b_n \in \mathbb{B}$ such that $b = \rho^{\mathcal{B}}(b_1, \ldots, b_n)$, the grammar $\Gamma_{h,C}$ has a rule $X^b \to \rho[X_1^{b_1}, \ldots, X_n^{b_n}]$; the axioms of $\Gamma_{h,C}$ are $\to X^c$, for each axiom $\to X$ of Γ and each element $c \in C$.

The grammars from the rest of the paper use the HR signature to describe parameterized systems. The following example shows two grammars that specify parameterized systems with chain-like and star-like network topologies, as in Figure 1 (b):

Example 4. The grammar Γ_{Chain} below defines systems with chain-like network topologies and at least three processes including the controller (Figure 1 (b) top):

$$\to C$$
$$C \to \mathsf{restrict}_{\{\sigma_1\}}((relC, get)_{(\sigma_3,\sigma_2)} \oplus (rel, get)_{(\sigma_2,\sigma_1)})$$
$$C \to \mathsf{restrict}_{\{\sigma_1\}}\mathsf{rename}_{\sigma_1 \leftrightarrow \sigma_2}(C \oplus (rel, get)_{(\sigma_1,\sigma_2)})$$

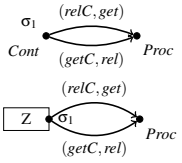


The right-hand sides of the last two rules correspond to the graphs on the right. Similarly, the Γ_{Star} grammar below defines systems with star-shaped network topology and at least two processes including the controller (Figure 1 (b) bottom):

$$\rightarrow Z$$

$$Z \rightarrow \mathsf{restrict}_{\{\sigma_1\}}((relC, get)_{(\sigma_1,\sigma_2)} \oplus^{\mathcal{S}} (getC, rel)_{(\sigma_1,\sigma_2)})$$

$$Z \rightarrow \mathsf{restrict}_{\{\sigma_1\}}(Z \oplus (relC, get)_{(\sigma_1,\sigma_2)} \oplus^{\mathcal{S}} (getC, rel)_{(\sigma_1,\sigma_2)})$$



The following argument will be repeated several times: if $\mathcal{A}$ is some HR algebra related to $\mathcal{S}$ by a homomorphism h, then $h(\mathcal{L}^{\mathcal{S}}(\Gamma)) = \mathcal{L}^{\mathcal{A}}(\Gamma)$, for each grammar Γ written using the HR signature. Moreover, if the domain of $\mathcal{A}$ is finite, the language $\mathcal{L}^{\mathcal{A}}(\Gamma)$ is finite and effectively computable by a finite Kleene iteration, provided that the interpretation of each function symbol from the HR signature in $\mathcal{A}$ is effectively computable. The finite and effectively computable algebras in question are circled in Figure 2.

As a consequence of Lemma 1, a grammar that specifies a parameterized system defines also its set of behaviors:

Lemma 2. *For each HR grammar Γ, we have $\beta(\mathcal{L}^{\mathcal{S}}(\Gamma)) = \mathcal{L}^{\mathcal{B}}(\Gamma)$.*

Proof. By Lemma 1, β is a homomorphism between $\mathcal{S}$ and $\mathcal{B}$, hence $\beta(\theta^{\mathcal{S}}) = \theta^{\mathcal{B}}$, for each ground term θ. The result follows, by the definition of the language of a grammar in a given algebra. □

We consider the problems of reachability and coverability for parameterized systems specified by grammars and give the formal definitions of the decision problems:

Definition 5. *The* $\mathsf{Reach}(\Gamma, Q, \mathsf{m})$ *(resp.* $\mathsf{Cover}(\Gamma, Q, \mathsf{m})$*) problem takes in input a grammar Γ, a set of places $Q \subseteq \mathsf{Q}_{\mathcal{P}}$, a mapping $\mathsf{m} : Q \rightarrow \mathbb{N}$ and asks for the existence of a system $\mathsf{S} \in \mathcal{L}^{\mathcal{S}}(\Gamma)$ and marking $\overline{\mathsf{m}} \in \mathsf{reach}(\beta(\mathsf{S}))$ (resp. $\overline{\mathsf{m}} \in \mathsf{cover}(\beta(\mathsf{S}))$) where $\sum_{v \in \mathsf{V_S} : q \in \mathsf{Q}_{\lambda_{\mathsf{S}}(v)}} \overline{\mathsf{m}}(q, v) = \mathsf{m}(q)$, for all $q \in Q$.*

Unsurprisingly, both problems are undecidable, even for simple grammars defining parameterized systems with chain-like topologies (see Example 4), when the structure of the process types is unrestricted.

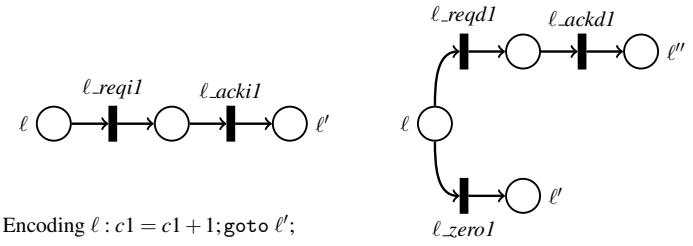
Theorem 2. *The* Reach *and* Cover *problems are undecidable.*

Proof. To show that the two mentioned problems are undecidable, we propose a reduction from the halting problem for a (two-counter) Minsky machine [**?**]. A Minsky machine is a finite sequence of instructions *Instrs* manipulating two natural variables $c1$ and $c2$, called counters, and each instruction has one of the following form:

1. $\ell : ci = ci + 1; \texttt{goto } \ell'$;
2. $\ell : \texttt{if } ci == 0 ? \texttt{ goto } \ell'; \texttt{else } ci = ci - 1; \texttt{goto } \ell''$;

where $\ell.\ell'$ and ℓ'' are labels. All labels are followed by an instruction except ℓ_f, the final label. The halting problem asks if the Minsky machine starting from an initial label ℓ_0 with the two counters set to 0 halts, *i.e.,* reaches the label ℓ_f.

We fix a Minsky machine given by its set of instructions *Instrs*. In order to simulate this model, we consider a family of systems which all have the following shape: there



(a) Encoding $\ell : c1 = c1 + 1; \texttt{goto}\ \ell'$;

(b) Encoding ℓ : $\texttt{if}\ c1 == 0 ?\ \texttt{goto}\ \ell'$;
$\texttt{else}\ c1 = c1 - 1; \texttt{goto}\ \ell''$;

Fig. 3. Simulating a Minsky machine with process type *Minsk*

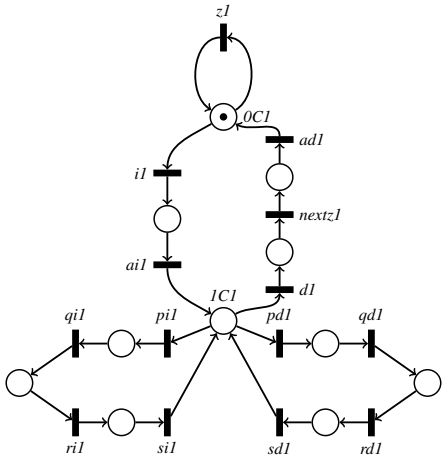


Fig. 4. Process type $Count_1$ to simulate the first counter

is a central vertex which simulates the instructions of the machine, on the right side of this vertex there is a sequence of vertices to encode the value of the counter $c1$ and on its left side a sequence of vertices to encode the value of the counter $c2$. The central node has process type *Minsk*, it is represented on Figure 3, where we notice that there is a place for each instruction label and furthermore the initial marking only puts a token in the place ℓ_0. The vertices to encode the value of counter $c1$ are labeled by the process type $Count_1$ depicted on Figure 4 and the vertices to encode the value of counter $c2$ are labeled the process type $Count_2$ which is similar to $Count_1$. We remark that in the process types $Count_1$, there are two specific places $0C1$ and $1C1$, and that initially the place $0C1$ is marked. During a simulation of the machine with a behavior, the number of vertices of process type $Count_1$ for which a token is in place $1C1$ corresponds to the current value of $c1$. To ease the explanation, we shall call the central vertex, the controller and the vertices labeled by $Count_i$ the counting processes for c_i, for $i = 1,2$. The grammar Γ_{Minsk} producing the desired family of systems is given on Figure 5 and an example of a system belonging to $\mathcal{L}^S(\Gamma_{Minsk})$ is depicted on Figure 6 (we omit the label on the edges to ease presentation).

We now provide the main ingredients of the simulation. At the beginning the controller is in state ℓ_0 and all the counting processes are in states $0C1$ or $0C2$ according to their process type, $Count_1$ or $Count_2$. At any 'big step' (*i.e.,* when forgetting intermediate steps needed for the correctness) of the simulation, when the controller is in a state corresponding to a label of the Minsky machine, the sequence of the counting processes of type $Count_1$ will always have the following form: it begins with a sequence of processes in state $1C1$ and ends with a sequence of processes in state $0C1$ (the same property holds for the counting processes of type $Count_2$ considering the state $1C2$ and $0C2$). Moreove, the number of processes in state $1C1$ (*resp.* $1C2$) represents the value of the first (*resp.* second) counter at this stage of the simulation. We have then the following behavior:

- when the controller simulates an instruction of the form $\ell : c1 = c1 + 1; \texttt{goto}\ \ell'$; it begins to request for an increment with the transition ℓ_reqi1 and it waits for its acknowledgment with ℓ_acki1. The request is then transmitted by all the processes in state $1C1$ to the first counting process in state $0C1$ which changes its state and acknowledges it moving to $1C1$, the acknowledgment being transmitted back to the controller. If there is no process of type $Count_1$ in state $0C1$, the simulation is stuck, it means that the system is not 'big' enough to simulate correctly the Minsky machine.
- when the controller simulates an instruction of the form $\ell : \texttt{if}\ c1 == 0? \texttt{goto}\ \ell'$; $\texttt{else}\ c1 = c1 - 1; \texttt{goto}\ \ell''$; and the counter $c1$ is equal to 0, this means that there is no counting process of type $Count_1$ in state $1C1$, then the controller takes the transition ℓ_zero1 together with the first counting process of type $Count_1$ which takes the transition $z1$.
- when the controller simulates an instruction of the form $\ell : \texttt{if}\ c1 == 0? \texttt{goto}\ \ell'$; $\texttt{else}\ c1 = c1 - 1; \texttt{goto}\ \ell''$; and the counter $c1$ has a strictly positive value, then, as for the increment, the controller requests a decrement with the transition ℓ_decqi1, this request is transmitted to the last counting process of type $Count_1$ in state $1C1$ (there is necessarily one) and if this last process has a right neighbor whose state is

$0C1$, then it moves to $0C1$ acknowledging the decrement, this acknowledgment being transmitted back to the controller which goes in state ℓ''. Note that each counting process in state $1C1$ can choose nondeterministically whether it is or not the last one in state $1C1$ by taking the transition $pd1$ (it transmits the decrement request) or $d1$ (it chooses it is the last one), but if it makes a bad choice, the simulation will be stuck.

– Finally, we have that the machine halts if and only if there is a system and an execution in its associated behavior where the controller ends in ℓ_f.

We now provide an example on how the transmission of the information is performed in a system. We consider a Minsky machine with two instructions $\ell_0 : c1 = c1+1;\texttt{goto}\ \ell'$; and $\ell' : c1 = c1+1;\texttt{goto}\ \ell''$;. Figure 7 presents a partial representation of a behavior simulating these instructions with two counter processes of type $Count_1$ (to make the figure readable, we put a dotted line between two transitions when they are synchronized). First, the controller simulates the instruction $\ell_0 : c1 = c1+1;\texttt{goto}\ \ell'$;, it takes the transition ℓ_0_reqi1 which says that it requests an increment, this transitions is 'synchronized' with two transitions $i1$ and $pi1$ of the first counting process on its right and since the state of this process is initially $0C1$ then this first process takes the transition $i1$. Afterwards, the controller waits for an acknowledgment of its request for an increment which will arrive when it takes the transition ℓ_0_acki1. This latter transition is synchronized with the two transitions ai_1 and si_1 of the first counting process. In our case, the pair (ℓ_0_acki1, ai_1) takes place and the controller ends in state ℓ' and the first counting process in $1C1$. The controller can now simulate $\ell' : c1 = c1+1;\texttt{goto}\ \ell''$;. It takes the transition ℓ'_reqi1, but this time the first counting process (which is in state $1C1$) takes the transition $pi1$ signifying it transmits the request to its neighbor (if there is one). Afterwards, if the first counting process has a neighbor, it fires the transition $qi1$ (which synchronizes with the transitions $i1$ and $pi1$ of its right neighbor), and its neighbor which is in state $0C1$ takes the transition $i1$, they then both move with the pair $(ri1, ai1)$ and the second counting process arrives in state $1C1$ whereas the first counting process is able to acknowledge the fact that the increment has been performed to the controller process with the synchronization (ℓ'_acki1, si_1). We end in a configuration where the controller is in state ℓ'' and the two first counting processes of type $Count_1$ are in state $1C1$.

We now present more formal arguments to explain why the reduction holds. We first describe the systems belonging to the language of the grammar Γ_{Minsk} (presented on Figure 5) in the algebras of systems. Given two naturals $m,n \geq 1$, we define the system $\mathsf{S}^{m,n}_{Minsk} = (\mathsf{V}, \mathsf{E}, \lambda)$ where:

– $\mathsf{V} = \{v^2_m, \ldots, v^2_1, v^c, v^1_1, \ldots, v^1_n\}$;
– $\mathsf{E} =$
$\{(v^2_i, a, v^2_{i-1}) \mid i \in [2,m] \text{ and } a \in \{(i2, qi2), (pi2, qi2), (ai2, ri2), (si2, ri2), (d2, qd2),$
$(pd2, qd2), (ad2, rd2), (sd2, rd2), (z2, nextz2)\}\}\cup$
$\{(v^1_i, a, v^1_{i+1}) \mid i \in [1, n-1] \text{ and } a \in \{(qi1, i1), (qi1, pi1), (ri1, ai1), (ri1, si1), (qd1, d1),$
$(qd1, pd1), (rd1, ad1), (rd1, sd1), (nextz1, z1)\}\}\cup$
$\{(v^2_1, a, v^c) \mid a \in \{(i2, \ell\_reqi2), (pi2, \ell\_reqi2), (ai2, \ell\_acki2), (si2, \ell\_acki2) \mid \ell \in L_{inc2}\}\}\cup$
$\{(v^2_1, a, v^c) \mid a \in \{(z2, \ell\_zero2), (d2, \ell\_reqd2), (pd2, \ell\_reqd2), (ad2, \ell\_ackd2), (sd2, \ell\_ackd2) \mid$
$\ell \in L_{dec2}\}\}\cup \{(v^c, a, v^1_1) \mid a \in \{(\ell\_reqi1, i1), (\ell\_reqi1, pi1), (\ell\_acki1, ai1), (\ell\_acki1, si1) \mid$

$$
\begin{array}{rcl}
 & \to & X_1 \\
X_1 & \to & X_2 \\
X_1 & \to & \mathsf{restrict}^{\mathcal{S}}_{\tau}(\mathsf{rename}^{\mathcal{S}}_{\alpha_1}(X_1 \oplus^{\mathcal{S}} Y_1)) \\
X_2 & \to & \mathit{Init} \\
X_2 & \to & \mathsf{restrict}^{\mathcal{S}}_{\tau}(\mathsf{rename}^{\mathcal{S}}_{\alpha_2}(Y_2 \oplus^{\mathcal{S}} X_2)) \\
\mathit{Init} & \to & \bigoplus^{\mathcal{S}}_{\ell \in L_{inc1}} \big((\ell\_reqi1, i1)^{\mathcal{S}}_{\sigma_0,\sigma_1} \oplus^{\mathcal{S}} (\ell\_reqi1, pi1)^{\mathcal{S}}_{\sigma_0,\sigma_1} \oplus^{\mathcal{S}} \\
 & & (\ell\_acki1, ai1)^{\mathcal{S}}_{\sigma_0,\sigma_1} \oplus^{\mathcal{S}} (\ell\_acki1, si1)^{\mathcal{S}}_{\sigma_0,\sigma_1}\big) \oplus^{\mathcal{S}} \\
 & & \bigoplus^{\mathcal{S}}_{\ell \in L_{dec1}} \big((\ell\_zero1, z1)^{\mathcal{S}}_{\sigma_0,\sigma_1} \oplus^{\mathcal{S}} \\
 & & (\ell\_reqd1, d1)^{\mathcal{S}}_{\sigma_0,\sigma_1} \oplus^{\mathcal{S}} (\ell\_reqd1, pd1)^{\mathcal{S}}_{\sigma_0,\sigma_1} \oplus^{\mathcal{S}} \\
 & & (\ell\_ackd1, ad1)^{\mathcal{S}}_{\sigma_0,\sigma_1} \oplus^{\mathcal{S}} (\ell\_ackd1, sd1)^{\mathcal{S}}_{\sigma_0,\sigma_1}\big) \oplus^{\mathcal{S}} \\
 & & \bigoplus^{\mathcal{S}}_{\ell \in L_{inc2}} \big((i2, \ell\_reqi2)^{\mathcal{S}}_{\sigma_2,\sigma_0} \oplus^{\mathcal{S}} (pi2, \ell\_reqi2)^{\mathcal{S}}_{\sigma_2,\sigma_0} \oplus^{\mathcal{S}} \\
 & & (ai2, \ell\_acki2)^{\mathcal{S}}_{\sigma_2,\sigma_0} \oplus^{\mathcal{S}} (si2, \ell\_acki2)^{\mathcal{S}}_{\sigma_2,\sigma_0}\big) \oplus^{\mathcal{S}} \\
 & & \bigoplus^{\mathcal{S}}_{\ell \in L_{dec2}} \big((z2, \ell\_zero2)^{\mathcal{S}}_{\sigma_2,\sigma_0} \oplus^{\mathcal{S}} \\
 & & (d2, \ell\_reqd2)^{\mathcal{S}}_{\sigma_2,\sigma_0} \oplus^{\mathcal{S}} (pd2, \ell\_reqd2)^{\mathcal{S}}_{\sigma_2,\sigma_0} \oplus^{\mathcal{S}} \\
 & & (ad2, \ell\_ackd2)^{\mathcal{S}}_{\sigma_2,\sigma_0} \oplus^{\mathcal{S}} (sd2, \ell\_ackd2)^{\mathcal{S}}_{\sigma_2,\sigma_0}\big) \\
Y_1 & \to & (qi1, i1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} (qi1, pi1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} \\
 & & (ri1, ai1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} (ri1, si1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} \\
 & & (qd1, d1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} (qd1, pd1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} \\
 & & (rd1, ad1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} (rd1, sd1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \oplus^{\mathcal{S}} \\
 & & (nextz1, z1)^{\mathcal{S}}_{\sigma_1,\sigma_3} \\
Y_2 & \to & (i2, qi2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} (pi2, qi2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} \\
 & & (ai2, ri2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} (si2, ri2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} \\
 & & (d2, qd2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} (pd2, qd2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} \\
 & & (ad2, rd2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} (sd2, rd2)^{\mathcal{S}}_{\sigma_4,\sigma_2} \oplus^{\mathcal{S}} \\
 & & (z2, nextz2)^{\mathcal{S}}_{\sigma_4,\sigma_2}
\end{array}
$$

$$
\begin{gathered}
\text{with } \mathrm{ptype}(\sigma_0) = \mathit{Minsk},\ \mathrm{ptype}(\sigma_1) = \mathrm{ptype}(\sigma_3) = \mathit{Count}_1,\ \mathrm{ptype}(\sigma_2) = \mathrm{ptype}(\sigma_4) = \mathit{Count}_2, \\
\alpha_1 = \{\sigma 1 \leftrightarrow \sigma 3\}, \\
\alpha_2 = \{\sigma 2 \leftrightarrow \sigma 4\}, \\
\tau = \{\sigma 1, \sigma 2\}, \\
L_{inci} = \{\ell \mid \ell : ci = ci + 1; \texttt{goto}\ \ell'; \in \mathit{Instrs}\} \text{ for } i \in \{1,2\} \\
L_{deci} = \{\ell \mid \ell : \texttt{if}\ ci == 0 ?\ \texttt{goto}\ \ell'; \texttt{else}\ ci = ci - 1; \texttt{goto}\ \ell''; \in \mathit{Instrs}\} \text{ for } i \in \{1,2\}
\end{gathered}
$$

Fig. 5. Grammar Γ_{Minsk} to produce systems for the simulation of a Minsky machine

$\ell \in L_{inc1}\}\}\cup$
$\{(v^c, a, v_1^1) \mid a \in \{(\ell\_zero1, z1), (\ell\_reqd1, d1), (\ell\_reqd1, pd1), (\ell\_ackd1, ad1), (\ell\_ackd1, sd1) \mid$
$\ell \in L_{dec1}\}\}$ where L_{inc1}, L_{dec1}, L_{inc2} and L_{dec2} are defined on Figure 5;

– $\lambda(v_i^2) = \mathit{Count}_2$ for all $i \in [1,m]$, $\lambda(v_i^1) = \mathit{Count}_1$ for all $i \in [1,n]$ and $\lambda(v^c) = \mathit{Minsk}$.

We can easily see that $\mathcal{L}^s(\Gamma_{\mathit{Minsk}}) = \{\mathsf{S}_{\mathit{Minsk}}^{m,n} \mid m \geq 1 \text{ and } n \geq 1\}$.

We have given on Figures 4 and 3, the way process types Count_1, Count_2 and Minsk are built. We notice that in the process type Minsk, there is a single vertex labelled by the halting label of the Minsky machine ℓ_f. We let $\mathsf{m}_f : Q \to \mathbb{N}$ (where Q is the set of states of the different processes) be the marking such that $\mathsf{m}_f(\ell_f) = 1$ and $\mathsf{m}_f(q) = 0$ for all other places q. Our reduction claims that the Minsky machine halts iff the answer to $\mathsf{Reach}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$ (*resp.* $\mathsf{Cover}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$) is positive.

Let us explain why the reduction holds for the two verification problems. First note that if the answer to $\mathsf{Reach}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$ is positive then so is the answer to $\mathsf{Cover}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$ by definition of the problem. Now assume the answer to $\mathsf{Cover}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$ is positive, it means that there exists a system $\mathsf{S}_{\mathit{Minsk}}^{m,n} = (\mathsf{V}, \mathsf{E}, \lambda)$ with $\mathsf{V} = \{v_m^2, \ldots, v_1^2, v^c, v_1^1, \ldots, v_n^1\}$ where v^c is the only vertex such that $\lambda(v^c) = \mathit{Minsk}$ and a marking $\overline{\mathsf{m}} \in \mathsf{cover}(\beta(\mathsf{S}_{\mathit{Minsk}}^{m,n}))$ verifying $\overline{\mathsf{m}}(\ell_f, v^c) = \mathsf{m}(\ell_f) = 1$. By definition, there exists hence a marking $\overline{\mathsf{m}}' \in \mathsf{reach}(\beta(\mathsf{S}_{\mathit{Minsk}}^{m,n}))$ such that $\overline{\mathsf{m}} \leq \overline{\mathsf{m}}'$. But since since $\beta(\mathsf{S}_{\mathit{Minsk}}^{m,n})$ is an automata-like PN, we have $\overline{\mathsf{m}}'(\ell_f, v^c) = 1$ and since v^c is the only vertex such that $\lambda(v^c) = \mathit{Minsk}$, we conclude that the answer to $\mathsf{Reach}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$ is positive.

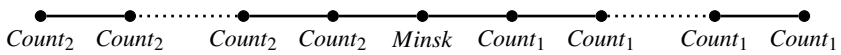


Fig. 6. Shape of a sytem for the simulation of a Minsky machine

Assume now that the Minsky machine halts. Let m and n be the maximum counter value taken by the counter $c2$ (*resp.* the counter $c1$) during the execution of the machine starting at ℓ_0 with $c1$ and $c2$ set at 0 and ending in ℓ_f. We can simulate this execution, as we have explained before, in the behavior $\beta(\mathsf{S}_{\mathit{Minsk}}^{m+1,n+1})$ to reach a marking with a token in the place (ℓ_f, v^c). As a matter of fact, the answer to $\mathsf{Cover}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$ is positive.

On the other side if the answer to $\mathsf{Cover}(\Gamma_{\mathit{Minsk}}, \{\ell_f\}, \mathsf{m}_f)$ is positive, there exists a system $\mathsf{S}_{\mathit{Minsk}}^{m,n} \in \mathcal{L}^s(\Gamma_{\mathit{Minsk}})$ and a marking $\overline{\mathsf{m}} \in \mathsf{reach}(\beta(\mathsf{S}_{\mathit{Minsk}}^{m,n}))$ such that $\overline{\mathsf{m}}(\ell_f, v^c) = 1$. The only way to reach such a marking from the initial marking of $\beta(\mathsf{S}_{\mathit{Minsk}}^{m,n})$ is to simulate faithfully at each step an instruction of the Minsky machine (if the behavior performs a wrong non-deterministic choice or if the vertices encoding the counters are not enough, the simulation get stuck and there is no way to put a token in a place of the process type Minsk corresponding to a label of the machine). We hence can rebuild an execution of the Minsky machine which ends in the label ℓ_f. □

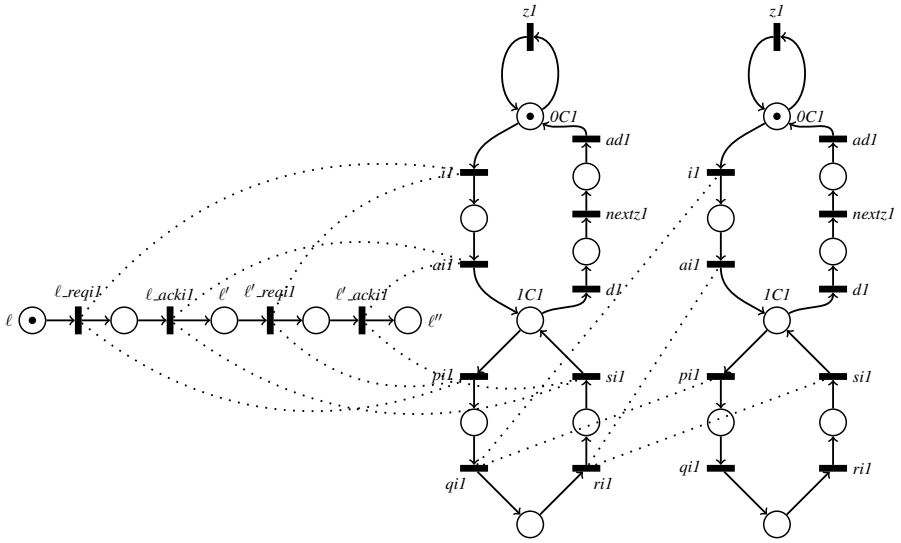


Fig. 7. Partial representation of a behavior for the simulation of a Minsky machine

3 Counting Abstraction

We define the *counting abstraction* of a set of behaviors as a set of PNs having finitely many underlying nets with possibly infinitely many initial markings each. The basic idea is to fold (i) the copies of the same place from some process type into a single place and (ii) the transitions having the same sets of predecessors and successors into a single transition. We show that the counting abstraction of the set of behaviors of a parameterized system described by an HR grammar can be computed by evaluating the same grammar in a finite HR algebra. Moreover, the infinite set of initial markings of a folded PN can be finitely described by another PN derived from the initial grammar describing the system. While the counting abstraction itself is not inherently tied to HR, and may abstract any family, the generation of initial markings on the other hand is strongly based on the HR-grammar (Section 3.2).

3.1 Folding

We define a folding function on the domain $\mathbb{B}$ of system behaviors. Let $\check{\mathsf{S}} = (\mathsf{S}, \xi)$ be a system and $(\mathcal{N}, \overline{\xi}) = \beta(\check{\mathsf{S}})$ be its behavior (Definition 3).

We recall that the places of $\mathcal{N}$ are pairs (q, v), where $q \in \mathsf{Q}_{\mathcal{P}}$, $v \in \mathsf{V_S}$ and the sources of the behavior are labeled by the function $\overline{\xi}$. A function $\eta : \Sigma \to \mathsf{V_S}$, having $\mathrm{dom}(\eta) \supseteq \mathrm{dom}(\xi)$, defines the following equivalence relation $\equiv_{[\eta]} \subseteq \mathsf{Q}_{\mathcal{N}} \times \mathsf{Q}_{\mathcal{N}}$:

$$(q_1, v_1) \equiv_{[\eta]} (q_2, v_2) \iff q_1 = q_2 \text{ and } \{v_1, v_2\} \cap \mathrm{img}(\eta) \neq \emptyset \Rightarrow v_1 = v_2 \qquad (1)$$

i.e., two places of $\mathcal{N}$ corresponding to the same place of a process type (within two distinct instances thereof) are considered equivalent, except for the sources with labels

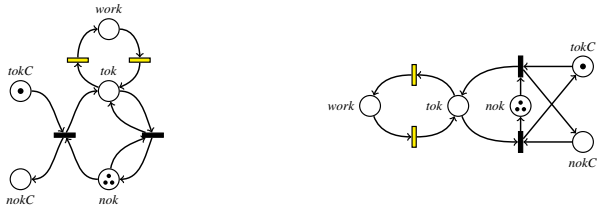


Fig. 8. Foldings of the behaviors given in Figure 1 (c) and (d), respectively.

η, that are equivalent only with themselves. The equivalence class of a place $(q,v) \in \mathsf{Q}_{\mathcal{N}}$ is denoted $[(q,v)]_{\equiv_{[\eta]}}$. Because we have assumed that the set of places corresponding to different process types are disjoint, $(q_1,v_1) \equiv_{[\xi]} (q_2,v_2)$ implies that $\lambda_{\mathsf{S}}(v_1) = \lambda_{\mathsf{S}}(v_2)$, *i.e.,* the two vertices are instances of the same process type.

The *folding* of $(\mathcal{N},\overline{\xi})$ is defined as $\phi(\mathcal{N},\overline{\xi}) \stackrel{\text{def}}{=} (\mathcal{N}_{/\equiv_{[\xi]}}, \overline{\xi}_{/\equiv_{[\xi]}})$, where, for each mapping $\eta : \Sigma \to \mathsf{V}_{\mathsf{S}}$ having $\mathrm{dom}(\eta) \supseteq \mathrm{dom}(\xi)$, we define $\overline{\xi}_{/\equiv_{[\eta]}}(\sigma,[q]_{\equiv_{[\eta]}}) \stackrel{\text{def}}{=} [\overline{\xi}(\sigma,q)]_{\equiv_{[\eta]}}$ if $\sigma \in \mathrm{dom}(\eta)$, else undefined, for each $\sigma \in \Sigma$. We refer to Figure 8 for examples.

The following lemma allows to define an algebra of abstract behaviors $\mathcal{B}^\sharp$ (Figure 2) from the algebra $\mathcal{B}$ of behaviors, using Proposition 1. The domain of $\mathcal{B}^\sharp$ is the set $\mathbb{B}^\sharp$ of folded behaviors, *i.e.,* quotients of behaviors $(\mathcal{N},\overline{\xi})$ w.r.t. the $\equiv_{[\xi]}$ relation.

Lemma 3. $\sim_{\ker(\phi)}$ *is an HR congruence.*

Proof. By a small abuse of notation we shall denote by $\equiv_{[\xi]}$ both the equivalence relation on places as well as the equivalence relation on transitions that it induces as defined in **??**.

The main proof relies on the following preliminary observations about manipulating quotients and markings, which hold for any permutation $\alpha : \Sigma \to \Sigma$ and any set $\tau \subseteq_{fin} \Sigma$:

(i) ξ and $\xi \circ \alpha^{-1}$ have the same range, which leads to $\equiv_{[\xi]}$ and $\equiv_{[\xi\circ\alpha^{-1}]}$ having the same equivalence classes and defining the same quotients;

(ii) the bijectivity of α and the fact that $\equiv_{[\xi]}$ does not identify vertices with different sources mean that $\overline{\xi \circ \alpha^{-1}} = \overline{\xi} \circ \alpha^{-1}$, and $(\overline{\xi} \circ \alpha^{-1})_{/\equiv_{[\xi]}} = \overline{\xi}_{/\equiv_{[\xi]}} \circ \alpha^{-1}$;

(iii) because $\xi\downarrow_\tau$ is a restriction of ξ, the equivalence relation induced by the former is a superset of the latter, and equivalence classes of $\equiv_{[\xi\downarrow_\tau]}$ are unions of equivalence classes of $\equiv_{[\xi]}$.

(iv) $\overline{\xi_1}_{/\equiv_{[\xi_1]}} = \overline{\xi_2}_{/\equiv_{[\xi_2]}}$ implies $\xi_1 = \xi_2$. Pick some $\overline{\xi_1}_{/\equiv_{[\xi_1]}}(\sigma,[q]_\sim) = \overline{\xi_2}_{/\equiv_{[\xi_2]}}(\sigma,[q]_\sim)$. Unfolding the definitions gives that $[\overline{\xi_1}(\sigma,q)]_{\equiv_{[\xi_1]}} = [\overline{\xi_2}(\sigma,q)]_{\equiv_{[\xi_2]}}$, which are equal to $[(q,\xi_1(\sigma))]_{\equiv_{[\xi_1]}}$ and $[(q,\xi_2(\sigma))]_{\equiv_{[\xi_2]}}$ respectively. The equivalence classes respect the labeling of sources, so both of the above places are alone in their equivalence class. This proves $(q,\xi_1(\sigma)) = (q,\xi_2(\sigma))$ from which we easily get $\xi_1(\sigma) = \xi_2(\sigma)$. Since the choice of $\sigma \in \Sigma$ was arbitrary, we obtain $\xi_1 = \xi_2$.

We now proceed separately for each HR function symbol:

– a_{σ_1,σ_2} is of arity zero thus trivially $a^{\mathcal{B}}_{\sigma_1,\sigma_2} \sim_{\ker(\phi)} a^{\mathcal{B}}_{\sigma_1,\sigma_2}$.

– rename_α: Let $(\mathcal{N}_1, \overline{\xi_1}) \sim_{\ker(\phi)} (\mathcal{N}_2, \overline{\xi_2})$ be two open behaviors having the same folded image. Using observation (i) above and given that $\mathsf{rename}_\alpha^{\mathcal{B}}$ does not modify the underlying net and the initial marking, we deduce that $\phi(\mathsf{rename}_\alpha^{\mathcal{B}}(\mathcal{N}_1, \overline{\xi_1}))$ and $\phi(\mathsf{rename}_\alpha^{\mathcal{B}}(\mathcal{N}_2, \overline{\xi_2}))$ have the same underlying net and initial marking. Then observation (ii) yields $(\overline{\xi_1 \circ \alpha^{-1}})_{/\equiv_{[\xi_1 \circ \alpha^{-1}]}} = \left(\overline{\xi_1}_{/\equiv_{[\xi_1]}}\right) \circ \alpha^{-1} = \left(\overline{\xi_2}_{/\equiv_{[\xi_2]}}\right) \circ \alpha^{-1} = (\overline{\xi_2 \circ \alpha^{-1}})_{/\equiv_{[\xi_2 \circ \alpha^{-1}]}}$ and we conclude that $\mathsf{rename}_\alpha^{\mathcal{B}}(\mathcal{N}_1, \overline{\xi_1}) \sim_{\ker(\phi)} \mathsf{rename}_\alpha^{\mathcal{B}}(\mathcal{N}_2, \overline{\xi_2})$
– $\mathsf{restrict}_\tau$: Let $(\mathcal{N}_1, \overline{\xi_1}) \sim_{\ker(\phi)} (\mathcal{N}_2, \overline{\xi_2})$ be two open behaviors having the same folded image. By observation (iii), we have that $\equiv_{[\xi_i]} \subseteq \equiv_{[\xi_i \downarrow \tau]}$ for $i = 1, 2$. This means that $\mathcal{N}_{i/\equiv_{[\xi_i \downarrow \tau]}} = \left(\mathcal{N}_{i/\equiv_{[\xi_i]}}\right)_{/\equiv_{[\xi_i \downarrow \tau]}}$ and because we know that (a) $\mathcal{N}_{1/\equiv_{[\xi_i]}} = \mathcal{N}_{2/\equiv_{[\xi_2]}}$ as well as (b) $\equiv_{[\xi_1 \downarrow \tau]} = \equiv_{[\xi_2 \downarrow \tau]}$, we deduce that $\mathcal{N}_{1/\equiv_{[\xi_1 \downarrow \tau]}} = \mathcal{N}_{2/\equiv_{[\xi_2 \downarrow \tau]}}$. Since, moreover, $\overline{\xi_1}_{/\equiv_{[\xi_1]}} = \overline{\xi_2}_{/\equiv_{[\xi_2]}}$, we conclude that $\overline{\xi_1}_{/\equiv_{[\xi_1 \downarrow \tau]}} = \overline{\xi_2}_{/\equiv_{[\xi_2 \downarrow \tau]}}$ and thus $\phi(\mathsf{restrict}_\tau^{\mathcal{B}}(\mathcal{N}_1, \overline{\xi_1})) = \phi(\mathsf{restrict}_\tau^{\mathcal{B}}(\mathcal{N}_2, \overline{\xi_2}))$.
– $\mathsf{op} = \oplus^{\mathcal{B}}$: Let $(\mathcal{N}_1, \overline{\xi_1}) \sim_{\ker(\phi)} (\mathcal{N}_2, \overline{\xi_2})$ and $(\mathcal{N}'_1, \overline{\xi_1}) \sim_{\ker(\phi)} (\mathcal{N}'_2, \overline{\xi_2})$ be pairwise equivalent behaviors. Lemma **??** gives us $\phi((\mathcal{N}_i, \xi_i) \oplus^{\mathcal{B}} (\mathcal{N}'_i, \xi_i)) = (\mathcal{N}''_{i/\equiv_{[\xi''_i]}}, \overline{\xi''_i}_{/\equiv_{[\xi''_i]}})$, where $\mathcal{N}''_i = (\mathcal{N}_i \uplus \mathcal{N}'_i)_{/\sim_{\ker(\overline{\xi_i}^{-1} \cup \overline{\xi'_i}^{-1})}}$ and $\xi''_i = (\overline{\xi_i} \cup \overline{\xi'_i})_{/\sim_{\ker(\overline{\xi_i}^{-1} \cup \overline{\xi'_i}^{-1})}}$. Since by the initial assumption we have $\overline{\xi_1}_{/\equiv_{[\xi_1]}} = \overline{\xi_2}_{/\equiv_{[\xi_2]}}$ we deduce by observation (iv) that $\xi_1 = \xi_2$. The same reasoning for ξ'_1 and ξ'_2 gives that the equivalence relations $\sim_{\ker(\overline{\xi_1}^{-1} \cup \overline{\xi'_1}^{-1})}$ and $\sim_{\ker(\overline{\xi_2}^{-1} \cup \overline{\xi'_2}^{-1})}$ are the same, hence $\xi''_1 = \xi''_2$.
It remains to prove that $\mathcal{N}''_{1/\equiv_{[\xi''_1]}} = \mathcal{N}''_{2/\equiv_{[\xi''_2]}}$. It helps that $\sim_{\ker(\overline{\xi_i}^{-1} \cup \overline{\xi'_i}^{-1})}$ fuses only places that are sources, while $\equiv_{[\xi''_i]}$ fuses only places that are not sources. This means that whenever a place is part of some non-singleton equivalence class according to $\equiv_{[\xi''_i]}$, its equivalence class according to $\sim_{\ker(\overline{\xi_i}^{-1} \cup \overline{\xi''_i}^{-1})}$ is a singleton, and reciprocally. Even though $\equiv_{[\xi''_i]}$ is technically defined on equivalence classes of places, this allows us to consider it as an equivalence relation between places, and to permute the two quotients to obtain $\mathcal{N}''_{i/\equiv_{[\xi''_i]}} = \left((\mathcal{N}_i \uplus \mathcal{N}'_i)_{/\equiv_{[\xi''_i]}}\right)_{/\sim_{\ker(\overline{\xi_i}^{-1} \cup \overline{\xi'_i}^{-1})}}$ It is now sufficient to prove $(\mathcal{N}_1 \uplus \mathcal{N}'_1)_{/\equiv_{[\xi''_1]}} = (\mathcal{N}_2 \uplus \mathcal{N}'_2)_{/\equiv_{[\xi''_2]}}$. Seeing that $\equiv_{[\xi_i]}, \equiv_{[\xi'_i]} \subseteq \equiv_{[\xi''_i]}$ we can insert quotients by $\equiv_{[\xi_i]}$ and $\equiv_{[\xi'_i]}$ underneath a quotient by $\equiv_{[\xi''_i]}$ without affecting the final result: $(\mathcal{N}_i \uplus \mathcal{N}'_i)_{/\equiv_{[\xi''_i]}} = \left(\mathcal{N}_{i/\equiv_{[\xi_i]}} \uplus \mathcal{N}'_{i/\equiv_{[\xi'_i]}}\right)_{/\equiv_{[\xi''_i]}}$. With this formulation it is finally clear that we can substitute the known equalities $\mathcal{N}_{1/\equiv_{[\xi_1]}} = \mathcal{N}_{2/\equiv_{[\xi_2]}}$ and $\mathcal{N}'_{1/\equiv_{[\xi'_1]}} = \mathcal{N}'_{2/\equiv_{[\xi'_2]}}$ to get the desired property.

Thus ϕ is an HR congruence. □

As an immediate consequence of Lemma 3, we obtain that ϕ is a homomorphism between $\mathcal{B}$ and $\mathcal{B}^\sharp$, hence the same HR grammar can be used to both specify an infinite set of behaviors and compute its folded abstraction:

Corollary 1. *For each HR grammar* Γ*, we have* $\phi(\mathcal{L}^{\mathcal{B}}(\Gamma)) = \mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)$.

Because $\mathcal{P}$ is a finite set of process types, each having finitely many places, the folded language $\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)$ has a finite set of underlying nets. This is because each transition t in a behavior $\mathcal{N} \in \mathcal{L}^{\mathcal{B}}(\Gamma)$ has at most two incoming/outgoing edges. Since the same holds for each transition of its quotient, *i.e.,* $\mathcal{N}_{/\equiv_{[\xi]}}$, there are finitely many places and transitions in each underlying net of some $\mathcal{N} \in \mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)$. Nevertheless, the language $\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)$ is unbounded, because the set of initial markings is unbounded. In order to represent this set in a finite way, we shall proceed in two steps:

1. Isolate the initial markings that correspond to a given net from $\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)$; we address this problem using the Filtering Theorem (Theorem 1).
2. Give a finite representation to the set of initial markings of each net; we tackle this problem using Esparza's idea [18] of building PNs that simulate the derivations of the "filtered" grammars obtained in the previous step.

To formally define the finite set of underlying nets from a language $\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)$, we consider the function ψ on the domain $\mathbb{B}^\sharp$, that drops the initial marking:

$$\psi((\mathsf{N}, \mathsf{m}_0), \xi) \stackrel{\mathsf{def}}{=} (\mathsf{N}, \xi) \tag{2}$$

Then, $\psi(\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma))$ is finite, because the numbers of places and transitions in each net from this set are bounded by $\|\mathsf{Q}_{\mathcal{P}}\|$ and $\|\mathsf{Q}_{\mathcal{P}}\|^4$ (*i.e.,* each transition has at most 2 incoming and 2 outgoing edges of weight 1), respectively.

Since none of the operations from $\mathcal{B}^\sharp$ modify the initial markings, it is straightforward that $\sim_{\ker(\psi)}$ is a HR congruence. We define the algebra $\mathcal{B}^\flat$ (Figure 2) having the finite domain $\mathbb{B}^\flat \stackrel{\mathsf{def}}{=} \psi(\mathbb{B}^\sharp)$ and the usual interpretations of the HR function symbols given by Proposition 1. By standard arguments (similar to Lemma 2 and Corollary 1), we obtain that ψ is a homomorphism between $\mathcal{B}^\sharp$ and $\mathcal{B}^\flat$, *i.e.,* $\psi(\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)) = \mathcal{L}^{\mathcal{B}^\flat}(\Gamma)$.

As previously discussed, $\mathcal{L}^{\mathcal{B}^\flat}(\Gamma)$ is a finite set. However, to ensure that this set can be effectively computed, the computation of the interpretation of the HR signature in $\mathcal{B}^\flat$ needs to be effective:

Proposition 2. *For each function symbol* op *from the HR signature, the function* $\mathsf{op}^{\mathcal{B}^\flat}$ *is effectively computable.*

Proof. Both the folding $\phi : \mathbb{B} \to \mathbb{B}^\sharp$ and the projection $\psi : \mathbb{B}^\sharp \to \mathbb{B}^\flat$ preserve the arity of transitions, so the elements of $\mathbb{B}^\flat$ obey $1 \leq \|{}^\bullet t\| = \|t^\bullet\| \leq 2$ for every transition t, and can thus be seen as behaviors with empty markings: we define $z : \mathbb{B}^\flat \to \mathbb{B}$ by $z((\mathsf{N}, \overline{\xi})) \stackrel{\mathsf{def}}{=} ((\mathsf{N}, 0), \overline{\xi})$. Because empty markings added together remain empty, $z(\mathbb{B}^\flat)$ forms a stable subset of $\mathbb{B}$ by all operations of HR. This homomorphism z is a right inverse of the translation from $\mathbb{B}$ to $\mathbb{B}^\flat$: $(\psi \circ \phi) \circ z = \mathrm{id}$.

By definition $\mathsf{op}^{\mathcal{B}^\flat}(b_1, ..., b_n) = (\psi \circ \phi)(\mathsf{op}^{\mathcal{B}}((\psi \circ \phi)^{-1}(b_1), ..., (\psi \circ \phi)^{-1}(b_n))$. Since $\sim_{\ker(\psi \circ \phi)}$ is an HR congruence, any right inverse of $\psi \circ \phi$ can replace $(\psi \circ \phi)^{-1}$ in the expression above, and a good candidate is z which is an effectively computable right inverse. This gives $\mathsf{op}^{\mathcal{B}^\flat}(b_1, ..., b_n) = \psi(\phi(\mathsf{op}^{\mathcal{B}}(z(b_1), ..., z(b_n))))$. All of z, ϕ, and ψ are effectively computable, so in order for $\mathsf{op}^{\mathcal{B}^\flat}$ to be effectively computable it suffices that

$\mathsf{op}^{\mathcal{B}}$ also be effectively computable, which is given by the alternative characterizations of HR operating directly on behaviors in Lemma **??**. □

By the previous arguments, the language $\mathcal{L}^{\mathcal{B}^\flat}(\Gamma)$ is finite and effectively computable, hence it can be produced by a finite Kleene iteration of the monotonic function that maps a tuple of sets indexed by the nonterminals of Γ into their $\mathcal{B}^\flat$ interpretations, given by the rules of Γ. Let $\{(\mathsf{N}_1,\overline{\xi_1}),\ldots,(\mathsf{N}_n,\overline{\xi_n})\} \stackrel{\mathsf{def}}{=} \mathcal{L}^{\mathcal{B}^\flat}(\Gamma)$ be this set. Using the Filtering Theorem (Theorem 1) one can effectively build grammars $\Gamma_1,\ldots,\Gamma_n$ such that:

$$\psi(\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma_i)) = \mathcal{L}^{\mathcal{B}^\flat}(\Gamma_i) = \{(\mathsf{N}_i,\overline{\xi_i})\}, \text{ for each } i \in [1,n] \tag{3}$$

More precisely, the Filtering Theorem gives, for each $i \in [1,n]$, a grammar Γ_i such that $\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma_i) = \mathcal{L}^{\mathcal{B}^\sharp}(\Gamma) \cap \psi^{-1}(\{(\mathsf{N}_i,\overline{\xi_i})\})$. By applying ψ to both sides of the equality, we obtain (Equation 3), thus taking care of the first step of the construction (1).

3.2 Initial Markings

Let $\Gamma = (\Xi,\Pi)$ be any of the grammars $\Gamma_1,\ldots,\Gamma_n$ (Equation 3). To simplify matters at hand, we assume w.l.o.g that the right-hand side of each rule in Γ has exactly one occurrence of an HR function symbol, *i.e.,* a constant a_{σ_1,σ_2}, a unary function symbol $\mathsf{restrict}_\tau$ or rename_α, or the binary function symbol $\oplus$, applied to 0, 1 or 2 variables, respectively. Note that each grammar can be put in this form, at the cost of adding polynomially many extra nonterminals.

First, we annotate each nonterminal $X \in \Xi$ with sets of sources τ that are *visible* (*i.e.,* have been introduced by a constant a_{σ_1,σ_2} and have not been removed by some application of $\mathsf{restrict}_{\tau'}$) in each complete derivation starting in X^τ, where X^τ is a shorthand for the pair (X,τ). The annotated grammar $\widehat{\Gamma} = (\widehat{\Xi},\widehat{\Pi})$ can be built from Γ by a standard worklist iteration [6] The relation between Γ and $\widehat{\Gamma}$ is captured below: Next, we use the annotated grammar $\widehat{\Gamma} = (\widehat{\Xi},\widehat{\Pi})$, to build a PN $\mathfrak{I}(\widehat{\Gamma})$ that generates the initial markings of Γ in the set of folded PNs (Corollary 1).

This construction is quite intuitive: by reinterpreting each rule of $\widehat{\Gamma}$ as a transition of $\mathfrak{I}(\widehat{\Gamma})$, we get a PN whose firing sequences mimick derivations of $\widehat{\Gamma}$ in which the rules/transitions of $\widehat{\Gamma}$ and $\mathfrak{I}(\widehat{\Gamma})$ are applied in the same order. More precisely, we create a Petri net with one place representing each nonterminal $X \in \widehat{\Xi}$, in which each rule of the form $X \to \rho[X_1,\ldots,X_n]$ is translated to some transition t such that ${}^\bullet t^\bullet = (X,\{X_1,\ldots,X_n\})$, and each rule of the form $\to X$ is translated to some transition t such that ${}^\bullet t^\bullet = (O,X)$ (here $O \notin \widehat{\Xi}$ is a newly created place that receives the initial token, as in Figure 9). This construction is such that a partial derivation having k occurrences of the nonterminal variable X, when reinterpreted as a firing sequence, leads to a marking with k tokens in the place corresponding to X. Assume that $X^\tau \Rightarrow^*_{\widehat{\Gamma}} \theta$ is a complete derivation, *i.e.,* θ is a ground HR term. Then, every instance of some process type p that occurs in the system $(\mathsf{S},\xi) \stackrel{\mathsf{def}}{=} \theta^{\mathcal{S}}$ starts in a vertex labeled by a source label $\sigma \in \Sigma$ such that, either:

(a) σ is removed by an application of $\mathsf{restrict}_{\tau'}$ such that $\sigma \notin \tau'$, then σ occurs in the subtree of θ rooted at the particular occurrence of $\mathsf{restrict}_{\tau'}$ that removed it, or

(b) σ is visible at the root of θ, *i.e.,* $\sigma \in \tau$.

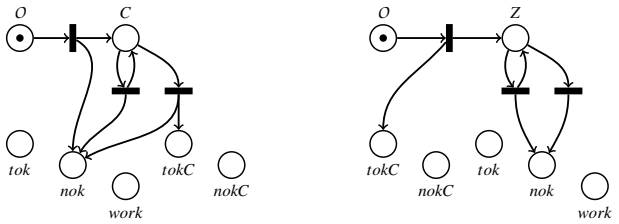


Fig. 9. $\mathfrak{I}(\widehat{\Gamma}_{Chain})$ (left) and $\mathfrak{I}(\widehat{\Gamma}_{Star})$ (right), showing how to initialize the marking of stars and chains generated by the grammars presented in Example 4.

When processing the rules of the form $X^{\tau} \to \mathsf{restrict}_{\tau'}(X_1{}^{\tau_1})$ and $\to X^{\tau}$ in the construction of $\mathfrak{I}(\widehat{\Gamma})$, we add an outgoing edge to each place q that is initially marked in $\mathrm{ptype}(\sigma)$, for some $\sigma \in \tau_1 \setminus \tau'$ or $\sigma \in \tau$. Then, for every instance of the process type that occurs in the system, its initial marking will be added to q by a firing sequence of $\mathfrak{I}(\widehat{\Gamma})$.

Example 5. Figure 9 shows the PNs obtained by this construction applied to $\widehat{\Gamma}_{Chain}$ and $\widehat{\Gamma}_{Star}$, *i.e.,* the annotated versions of the two grammars from Example 4. Intuitively, a chain system generated by the grammar Γ_{Chain}, see Figure 1 (b) top, has 1 instance of the process type *Cont* and $n \geq 2$ instances of the process type *Proc*. This means that, initially, there is 1 token on the place *tokC* and $n \geq 2$ tokens in *nok*, as in Figure 9 (a) Similarly, the initial marking of a star system, see Figure 1 (b) bottom, generated by the grammar Γ_{Star} has 1 token in *tokC* and $n \geq 1$ tokens in *nok*, as in Figure 9 (b).

We illustrate the construction of $\mathfrak{I}(\widehat{\Gamma}_{Chain})$ by explaining how the transitions labeled (1) and (2) in Figure 9 have been introduced:

(1) corresponds to the rule $\to C^{\{\sigma_1\}}$: the final system has one visible source label σ_1, whose process type is $\mathrm{ptype}(\sigma_1) = Proc$. When this rule is applied, it creates one instance of *Proc*, hence the outgoing edge to *nok*. Since this rule is, its corresponding transition has an incoming edge from the start place O of $\mathfrak{I}(\widehat{\Gamma}_{Chain})$ and produces an instance of $C^{\{\sigma_1\}}$, hence the outgoing edge to C.

(2) corresponds to the rule $C^{\{\sigma_1\}} \to \mathsf{restrict}_{\{\sigma_1\}}\mathsf{rename}_{\sigma_1 \leftrightarrow \sigma_2}(C^{\{\sigma_1\}} \oplus (rel, get)_{(\sigma_1,\sigma_2)})$: it consumes one occurrence of $C^{\{\sigma_1\}}$ and produces another, hence having both an incoming and outgoing edge to C. The restrict operation hides a σ_2-source, where $\mathrm{ptype}(\sigma_2) = Proc$, thus each application of this rule is responsible for one additional instance of the process type *Proc*.

For a PN $\mathcal{N}$ and a set of places $Q \subseteq \mathsf{Q}_{\mathcal{N}}$, we denote by:

$$\mathrm{reach}^0_Q(\mathcal{N}) \stackrel{\mathsf{def}}{=} \{\mathsf{m} \in \mathrm{reach}(\mathcal{N}) \mid \mathsf{m}(q) = 0, \text{ for all } q \in Q\} \tag{4}$$

the set of reachable markings having zero tokens in a place from Q. For a set $\mathcal{M}$ of markings and a set Q of places, we denote by $\mathcal{M}\downarrow_Q$ the set of restrictions of each $\mathsf{m} \in \mathcal{M}$ to the places in Q. The relation between the set of folded PNs described by an annotated grammar $\widehat{\Gamma}$ and the PN $\mathfrak{I}(\widehat{\Gamma})$ is formally captured below:

[6] The annotation algorithm is given in the full version of the paper. [10]

Lemma 4. *Let $\widehat{\Gamma} = (\widehat{\Xi}, \widehat{\Pi})$ be an annotated grammar. Then, we have:*

$$\{\mathrm{init}_{\mathcal{N}} \mid (\mathcal{N}, \xi) \in \mathcal{L}^{\mathcal{B}^\sharp}(\widehat{\Gamma})\} = \mathrm{reach}^0_{\{O\} \uplus \widehat{\Xi}}(\mathfrak{I}(\widehat{\Gamma})) \downharpoonright_{\mathsf{Q}_{\mathcal{P}}}$$

Proof. Let $\mathfrak{I}(\widehat{\Gamma}) \stackrel{\mathsf{def}}{=} (\mathsf{N}, \mathsf{m}_0)$ in the rest of this proof.
"$\subseteq$" Let $(\mathcal{N}, \xi) \in \mathcal{L}^{\mathcal{B}^\sharp}(\widehat{\Gamma})$ be a folded PN and $\mathsf{m} \stackrel{\mathsf{def}}{=} \mathrm{init}_{\mathcal{N}}$ be its inital marking. By the definition of the $\mathcal{B}^\sharp$ algebra, there exists an open behavior $(\overline{\mathcal{N}}, \overline{\xi}) \in \mathcal{L}^{\mathcal{B}}(\widehat{\Gamma})$ such that $\phi(\overline{\mathcal{N}}, \overline{\xi}) = (\mathcal{N}, \xi)$. Let $X^\tau \Rightarrow^*_{\widehat{\Gamma}} \theta$ be a complete derivation, starting with an axiom $\rightarrow X^\tau \in \widehat{\Pi}$ and ending with a ground term θ, such that $\theta^{\mathcal{B}} = (\overline{\mathcal{N}}, \overline{\xi})$. Let $\overline{\mathsf{m}} \stackrel{\mathsf{def}}{=} \mathrm{init}_{\overline{\mathcal{N}}}$ be the initial marking of $\overline{\mathcal{N}}$. Note that $\overline{\mathsf{m}}(q, v) = 1$ if q is initially marked in its process type and $\overline{\mathsf{m}}(q, v) = 0$ otherwise, for all $(q, v) \in \mathsf{Q}_{\overline{\mathcal{N}}}$. Then $\mathsf{m}(q) = \sum_{(q,v) \in \mathsf{Q}_{\overline{\mathcal{N}}}} \overline{\mathsf{m}}(q, v)$, for all $q \in \mathsf{Q}_{\mathcal{P}}$ (we recall that the $\equiv_{[\overline{\xi}]}$-equivalence classes are denoted by places from the process types). Let $\mathsf{m}_0 \stackrel{\mathbf{t}}{\rightsquigarrow} \mathsf{m}'$ be the firing sequence of $\mathfrak{I}(\widehat{\Gamma})$ that mimicks the derivation $X^\tau \Rightarrow^*_{\widehat{\Gamma}} \theta$. By the definition of $\mathfrak{I}(\widehat{\Gamma})$ we have $\mathsf{m}' \in \mathrm{reach}(\mathfrak{I}(\widehat{\Gamma}))$ and $\mathsf{m}'(x) = 0$, for all $x \in \{O\} \uplus \widehat{\Xi}$. It remains to prove that m' and m agree over $\mathsf{Q}_{\mathcal{P}}$. Let $q \in \mathsf{Q}_{\mathcal{P}}$ be a place and distinguish the following cases:

- If q is not initially marked in its process type then $\mathsf{m}(q) = 0$. But $\mathsf{m}'(q) = 0$ follows from the definition of $\mathfrak{I}(\widehat{\Gamma})$, because the only places from $\mathsf{Q}_{\mathcal{P}}$ having an incoming edge in $\mathfrak{I}(\widehat{\Gamma})$ are the places from $\mathsf{Q}_{\mathcal{P}}$ that are initially marked in their respective process types.
- Else, q is initially marked in its process type and $\mathsf{m}(q)$ is the number of instances of that process type in the system $\theta^{\mathcal{S}}$. Then, $\mathsf{m}'(q) = \mathsf{m}(q)$ by the argument made at points (a) and (b) above.

"$\supseteq$" Let $\mathsf{m} \in \mathrm{reach}(\mathfrak{I}(\widehat{\Gamma}))$ be a marking such that $\mathsf{m}(x) = 0$, for all $x \in \{O\} \uplus \widehat{\Xi}$. Since each firing sequence of $\mathfrak{I}(\widehat{\Gamma})$ starting from m_0 corresponds to a derivation of $\widehat{\Gamma}$ starting from an axiom $\rightarrow X^\tau \in \widehat{\Pi}$, the firing sequence $\mathsf{m}_0 \stackrel{\mathbf{t}}{\rightsquigarrow} \mathsf{m}$ corresponds to a complete derivation $X^\tau \Rightarrow^*_{\widehat{\Gamma}} \theta$, because $\mathsf{m}(x) = 0$, for all $x \in \{O\} \uplus \widehat{\Xi}$, by the definition of $\mathfrak{I}(\widehat{\Gamma})$. Let $(\overline{\mathcal{N}}, \overline{\xi}) = \theta^{\mathcal{B}}$ be the behavior of the system built by this derivation and $\overline{\mathsf{m}} \stackrel{\mathsf{def}}{=} \mathrm{init}_{\overline{\mathcal{N}}}$ be its initial marking. Let $(\mathcal{N}, \xi) \stackrel{\mathsf{def}}{=} \phi(\overline{\mathcal{N}}, \overline{\xi}) = \theta^{\mathcal{B}^\sharp}$ be the folded PN of this behavior and $\mathsf{m}' \stackrel{\mathsf{def}}{=} \mathrm{init}_{\mathcal{N}}$ be its initial marking, *i.e.,* $\mathsf{m}'(q) = \sum_{(q,v) \in \mathsf{Q}_{\overline{\mathcal{N}}}} \overline{\mathsf{m}}(q, v)$, for all $q \in \mathsf{Q}_{\mathcal{P}}$. We prove that m and m' agree over $\mathsf{Q}_{\mathcal{P}}$, by distinguishing the cases below:

- If q is not initially marked in its process type then $\mathsf{m}'(q) = 0$. In this case $\mathsf{m}(q) = 0$ because there are no incoming edges to q in $\mathrm{init}_{\widehat{\Gamma}}$, by the definition of the latter.
- Else, q is initially marked in its process types and $\mathsf{m}'(q)$ is the number of instances of that process type in the system $\theta^{\mathcal{S}}$. Then, $\mathsf{m}(q) = \mathsf{m}'(q')$, by the argument made at points (a) and (b) above. □

□

3.3 Soundness

We now have all the elements to describe our counting abstraction method and prove its soundness, *i.e.,* if the reachability (resp. coverability) problem has a negative answer for the abstraction, then the concrete reachability (resp. coverability) problem has a negative answer.

For each grammar $\Gamma_i = (\Xi_i, \Pi_i)$ defined at (Equation 3), that corresponds to the (open) net (N_i, ξ_i), for $i \in [1,n]$, we define the PN $\mathfrak{N}(\Gamma_i) \stackrel{\text{def}}{=} (\mathcal{N}_i, \xi_i)$, where:

$$\mathsf{Q}_{\mathcal{N}_i} \stackrel{\text{def}}{=} \mathsf{Q}_{\mathfrak{I}(\widehat{\Gamma}_i)} \cup \mathsf{Q}_{\mathsf{N}_i} \quad \mathsf{T}_{\mathcal{N}_i} \stackrel{\text{def}}{=} \mathsf{T}_{\mathfrak{I}(\widehat{\Gamma}_i)} \uplus \mathsf{T}_{\mathsf{N}_i} \quad \mathsf{W}_{\mathcal{N}_i} \stackrel{\text{def}}{=} \mathsf{W}_{\mathfrak{I}(\widehat{\Gamma}_i)} \cup \mathsf{W}_{\mathsf{N}_i} \quad \text{init}_{\mathcal{N}_i} \stackrel{\text{def}}{=} \text{init}_{\mathfrak{I}(\widehat{\Gamma}_i)}$$

Note that $\text{reach}^0_{\mathsf{Q}_{\mathfrak{I}(\widehat{\Gamma}_i)} \setminus \mathsf{Q}_{\mathsf{N}_i}}(\mathfrak{N}(\Gamma_i))$ is the set of markings of $\mathfrak{N}(\Gamma_i)$ that can be reached *after* the full generation of its initial marking, *i.e.,* from those markings that have no more tokens in any of the places of $\mathfrak{I}(\widehat{\Gamma}_i)$, excepted the initially marked places of $\mathsf{Q}_{\mathcal{P}}$.

Our verification method for the grammar-based parameterized reachability and coverability problems (Definition 5) relies on the construction of a finite number of PNs $\mathfrak{N}(\Gamma_1), \ldots, \mathfrak{N}(\Gamma_n)$ from a given grammar Γ that describes a set of systems. The relation between the reachability (resp. cover) set of the original parameterized system and its finitary abstraction is captured by the following lemma:

The soundness of the method is formally captured below:

Theorem 3. *Let Γ be an HR grammar such that $\mathcal{L}^{\mathcal{B}^\flat}(\Gamma) = \{(\mathsf{N}_1, \xi_1), \ldots, (\mathsf{N}_n, \xi_n)\}$, $Q \subseteq \mathsf{Q}_{\mathcal{P}}$ a set of places, $\mathsf{m} : Q \to \mathbb{N}$ a mapping. Then, one can effectively build grammars $\Gamma_1, \ldots, \Gamma_n$ such that $\mathcal{L}^{\mathcal{B}^\flat}(\Gamma_i) = \{(\mathsf{N}_i, \xi_i)\}$, for all $i \in [1,n]$ and:*

1. $\mathsf{Reach}(\Gamma, Q, \mathsf{m})$ *has a negative answer if* $\mathsf{m} \notin \bigcup_{i=1}^n \text{reach}^0_{\mathsf{Q}_{\mathfrak{N}(\Gamma_i)} \setminus \mathsf{Q}_{\mathcal{P}}}(\mathfrak{N}(\Gamma_i)) \downarrow_Q$.
2. $\mathsf{Cover}(\Gamma, Q, \mathsf{m})$ *has a negative answer if* $\mathsf{m} \notin \bigcup_{i=1}^n \text{cover}(\mathfrak{N}(\Gamma_i)) \downarrow_Q$.

Proof. The effective construction of $\Gamma_1, \ldots, \Gamma_n$ follows from Proposition 2 and the effectiveness of the Fitering Theorem (Theorem 1).

(1) We prove the contrapositive: $\mathsf{Reach}(\Gamma, Q, \mathsf{m})$ has a positive answer

$\iff \exists \mathsf{S} \in \mathcal{L}^{\mathcal{S}}(\Gamma)\ \exists \overline{\mathsf{m}} \in \text{reach}(\beta(\mathsf{S}))\ \forall q \in Q\,.\, \sum_{v \in \mathsf{V}_\mathsf{S} \mid q \in \mathsf{Q}_{\lambda_\mathsf{S}(v)}} \overline{\mathsf{m}}(q) = \mathsf{m}(q)$, by definition of Reach

$\iff \exists (\mathcal{N}, \xi) \in \mathcal{L}^{\mathcal{B}}(\Gamma)\ \exists \overline{\mathsf{m}} \in \text{reach}(\mathcal{N})\ \forall q \in Q\,.\, \sum_{v \in \mathsf{V}_\mathsf{S} \mid q \in \mathsf{Q}_{\lambda_\mathsf{S}(v)}} \overline{\mathsf{m}}(q) = \mathsf{m}(q)$, by Proposition 2

$\iff \exists (\mathcal{N}, \xi) \in \mathcal{L}^{\mathcal{B}}(\Gamma)\ \exists \mathsf{m}' \in \text{reach}(\mathcal{N}_{/\equiv_{[\xi]}})\,.\, \mathsf{m}' \downarrow_Q = \mathsf{m}$

$\implies \exists (\mathcal{N}, \xi) \in \mathcal{L}^{\mathcal{B}}(\Gamma)\ \exists \mathsf{m}' \in \text{reach}(\mathcal{N}_{/\equiv_{[\xi]}})\,.\, \mathsf{m}' \downarrow_Q = \mathsf{m}$, by Lemma **??**, **??**

$\iff \mathsf{m} \in \text{reach}(\mathcal{L}^{\mathcal{B}^\sharp}(\Gamma)) \downarrow_Q$, by Proposition 1

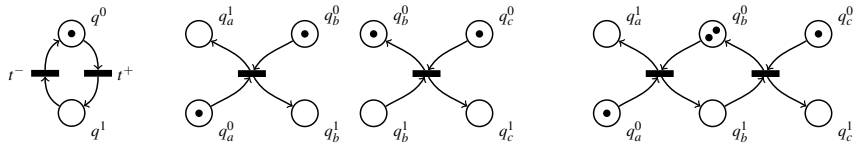
$\iff \mathsf{m} \in \bigcup_{i=1}^n \left(\text{reach}^0_{\mathsf{Q}_{\mathfrak{N}(\Gamma_i)} \setminus \mathsf{Q}_{\mathcal{P}}}(\mathfrak{N}(\Gamma_i)) \right) \downarrow_{\mathsf{Q}_{\mathcal{P}}}$, by Lemma **??**, **??**

(2) Along the same lines as (1), using Lemma **??**, **??** instead of **??**. □

Note that, if $\mathsf{Reach}(\Gamma, Q, \mathsf{m})$ (*resp.* $\mathsf{Cover}(\Gamma, Q, \mathsf{m})$) has a negative answer, then each instance the parameterized system described by Γ (*i.e.,* the set of systems $\mathcal{L}^S(\Gamma)$) is safe with respect to the property encoded by the marking m, *i.e.,* does not reach (*resp.* cover) the marking m over the set of places Q. For instance, mutual exclusion (only one process of a certain type in a certain state) is naturally encoded as a coverability problem.

3.4 False Positives

As we have announced, our abstraction is sound but not complete. Before we explore a decidable fragment, we show here a simple example of a net on which false positives appear (*i.e.,* the abstraction exhibits a violation of safety, but the original net is safe).



(a) $\mathsf{p}_a, \mathsf{p}_b, \mathsf{p}_c$ (b) $\theta^{\mathcal{B}}$ is safe (c) $\theta^{\mathcal{B}^\sharp}$ is not safe

Fig. 10. 3 process types with the same net, and a way of connecting them that produces a false positive when folded. $\theta \stackrel{\text{def}}{=} (t^+,t^+)_{\sigma_a,\sigma_b} \oplus (t^-,t^+)_{\sigma'_b,\sigma_c}$ with $\mathrm{ptype}(\sigma_a) = \mathsf{p}_a$, $\mathrm{ptype}(\sigma_b) = \mathrm{ptype}(\sigma'_b) = \mathsf{p}_b$, $\mathrm{ptype}(\sigma_c) = \mathsf{p}_c$. The safety property is $\mathsf{m}_{\mathsf{tgt}}(q_c^1) \geq 1$.

Figure 10 illustrates one phenomenon that is a possible cause of false positives. Here, the folding changes the connectivity of the network: two instances of p_b are folded together and make a path from p_a to p_c appear that was absent from the original system. The coverability query $\mathsf{m}_{\mathsf{tgt}}(q_c^1) \geq 1$ is unsatisfiable in $\theta^{\mathcal{B}}$, but satisfiable in $\theta^{\mathcal{B}^\sharp}$.

To enable the verification of infinite systems that exhibit a similar issue, here is one possible approach. We give ourselves a new process type p'_b, identical copy of p_b, and we alter the grammar and/or the assignment $\mathrm{ptype}(\cdot)$ to distinguish between the instances we do not want folded together. Here we could change $\mathrm{ptype}(\sigma'_b) = \mathsf{p}_{b'}$ (instead of p_b). The same abstraction applied to the new system will not fold together instances of p_b with instances of p'_b, making the false positive disappear. In general on infinite families, we may modify the grammar to select a subset of instances of a process type to instead use a copy of the process type that is folded separately. Repeating this process finitely many times keeps the abstraction finite. This refinement technique constitute what we call a *partial unfolding*, because we select a few instances to not be folded with the rest. Automated detection of when this technique is applicable, and more advanced transformations of the grammar constitute a research avenue orthogonal to the abstraction technique we explore here, and may be the topic of future work.

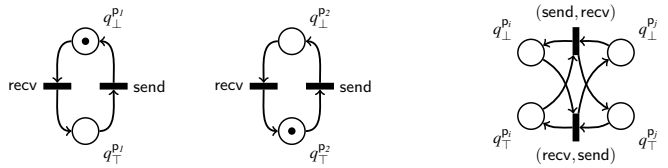


Fig. 11. Two process types (left), and two kinds of interactions (right), which all other process types and interactions in our restriction have the same shape as.

4 A Decidable Fragment

We have shown that the parameterized coverability problem is undecidable, for systems with fairly simple network topologies (chains) and unrestricted process types (Theorem 2). We refine this result by proving that only restricting the process types, but not the topology of the network, suffices to recover decidability. This approach is inspired by [5], in which *token-passing systems* (TPS) are found to admit cutoffs, and thus decidable model-checking, provided that the architecture is definable in monadic second order logic and of bounded tree-width, like ours. TPS and PPS are similar, with the tradeoff that TPS may have internal transitions, but only one token in the system.

Albeit based on a simple communication pattern (*i.e.,* passing a pebble from one node to a neighbour having no pebble), our decidable fragment is non-trivial: we found it to be in 2EXPTIME, with a PSPACE-hard lower bound.

4.1 Pebble-Passing Systems

The class of pebble-passing systems (PPS) is defined by restricting the process types and interactions of a system, as in Figure 11. The example from Figure 10 also happens to be of this form. We give the formal definition below:

Definition 6. *Let $\mathcal{P}_{\mathsf{pps}}$ be a set of process types, where $\mathsf{Q}_\mathsf{p} = \{q^\mathsf{p}_\bot, q^\mathsf{p}_\top\}$ and $\mathsf{T}_\mathsf{p} = \mathsf{T}^{obs}_\mathsf{p} = \{\mathsf{send}, \mathsf{recv}\}$, such that ${}^\bullet\mathsf{send}^\bullet = (q^\mathsf{p}_\top, q^\mathsf{p}_\bot)$ and ${}^\bullet\mathsf{recv}^\bullet = (q^\mathsf{p}_\bot, q^\mathsf{p}_\top)$, for each $\mathsf{p} \in \mathcal{P}_{\mathsf{pps}}$. Let $\Delta_{\mathsf{pps}} \stackrel{\mathsf{def}}{=} \{(\mathsf{send}, \mathsf{recv}), (\mathsf{recv}, \mathsf{send})\}$ be a set of edge labels. A system $\mathsf{S} = (\mathsf{V}, \mathsf{E}, \lambda)$ over $\mathcal{P}_{\mathsf{pps}}$ and Δ_{pps} is said to be* pebble-passing.

Intuitively, a token in $q^\mathsf{p}_\top$ (*i.e.,* $\mathsf{m}(q^\mathsf{p}_\top) = 1$) represents the ownership of a ressource, called *pebble*, and a token in $q^\mathsf{p}_\bot$ is the absence of a pebble, called *hole*. Since each process type is automata-like (*i.e.,* has a token in exactly one place), each node of the system can have either a pebble or a hole, in all the reachable markings of its behavior (Definition 3). An edge $(v, (\mathsf{send}, \mathsf{recv}), v')$ (*resp.* $(v, (\mathsf{recv}, \mathsf{send}), v')$) will be denoted $v \to v'$ (*resp.* $v' \to v$). Intuitively, firing an interaction $v \to v'$ moves a pebble from v to v' and simultaneously moves a hole from v' to v. Thus each transition preserves the total numbers of pebbles and holes in the system, respectively.

Pebble-passing systems have very strict constraints on their process types and interactions, but no constraints on the set of network topologies, other than that it is definable by an HR grammar written with constants of the form $(\mathsf{send}, \mathsf{recv})_{\sigma_1,\sigma_2}$ or $(\mathsf{recv}, \mathsf{send})_{\sigma_1,\sigma_2}$, for some source labels $\sigma_1, \sigma_2 \in \Sigma$, where Σ is the set of source labels

that may occur in a grammar. We denote by $\mathsf{HR}_{\mathsf{pps}}$ the signature of these grammars. The rest of this section is concerned with the proof of the following theorem:

Theorem 4. *The* Cover *problem for grammars written using the* $\mathsf{HR}_{\mathsf{pps}}$ *signature is in 2EXPTIME and PSPACE-hard.*

The proof of Theorem 4 is organized as follows. The double exponential upper bound relies on a result showing that, in order to cover a target marking $\mathsf{m}_{\mathsf{tgt}}$ it is sufficient to consider only firing sequences that cross (*i.e.,* move a pebble to and from) each place at most K times, where K is the size of the unary encoding of $\mathsf{m}_{\mathsf{tgt}}$. Based on this result (Lemma 6), we define an effectively computable algebra $\mathcal{F}$ (Figure 2), having the property that the coverability problem reduces to a membership test on the language of the input grammar in $\mathcal{F}$. Here the use of HR guarantees that $\mathcal{F}$ is finite. The upper bound follows from the fact that $\mathcal{L}^{\mathcal{F}}(\Gamma)$ is computable in double exponential time (see Proposition 3 for a precise estimation). The lower bound uses a polynomial reduction from the emptiness problem for 2-way nondeterministic automata [21] (2NFA). This construction works because (1) 2NFAs run on words which have a chain-like and thus HR-expressible structure, and (2) 2NFAs have only finite memory, which we are able to encode within the constraints of PPS.

4.2 Firing Sequences

For the rest of this section, let $\check{\mathsf{S}} = (\mathsf{S}, \xi)$ be a fixed open pebble-passing system, having an underlying system $\mathsf{S} = (\mathsf{V}, \mathsf{E}, \lambda)$ whose behavior is $\beta(\mathsf{S}) \stackrel{\text{def}}{=} (\mathsf{N}, \mathsf{m}_0)$. Below, we introduce an equivalent characterization of the firing sequences of $\beta(\mathsf{S})$.

The *footprint* of a marking m is a mapping $\mathsf{fp}_{\mathsf{m}} : \mathsf{V} \to \{0,1\}$ defined as $\mathsf{fp}_{\mathsf{m}}(v) \stackrel{\text{def}}{=} \mathsf{m}(q_{\top}^{\lambda(v)}, v)$, for each vertex $v \in \mathsf{V}$. Note that fp_{m} evaluates to 1 on pebble and to 0 on hole vertices. We say that a marking footprint π is *valid over* $\mathcal{V} \subseteq \mathsf{V}$ iff $0 \leq \pi(v) \leq 1$ for each vertex $v \in \mathcal{V}$, *resp. valid*, when $\mathcal{V} = \mathsf{V}$ follows from the context.

Given a subset of states $Q \subseteq \mathsf{Q}_{\mathcal{P}}$ and a marking to cover $\mathsf{m}_{\mathsf{tgt}} : Q \to \mathbb{N}$, the coverability problem asks for the existence of a reachable marking $\mathsf{m} : \mathsf{Q}_{\mathsf{N}} \to \{0,1\}$ such that, for each process type $\mathsf{p} \in \mathcal{P}$ and each place $q \in Q$:

$$\|\{v \in \lambda^{-1}(\mathsf{p}) \mid \mathsf{fp}_{\mathsf{m}}(v) = 0\}\| \geq \mathsf{m}_{\mathsf{tgt}}(q_{\bot}^{\mathsf{p}}) \tag{5}$$

$$\|\{v \in \lambda^{-1}(\mathsf{p}) \mid \mathsf{fp}_{\mathsf{m}}(v) = 1\}\| \geq \mathsf{m}_{\mathsf{tgt}}(q_{\top}^{\mathsf{p}}) \tag{6}$$

The footprint $\mathsf{fp}_{v \to v'} : \mathsf{V} \to \mathbb{Z}$ of an edge $v \to v' \in \mathsf{E}$ is defined as $\mathsf{fp}_{v \to v'}(u) \stackrel{\text{def}}{=} 1$ if $u = v$, -1 if $u = v'$ and 0, otherwise. We extend footprints to sequences of edges $\mathbf{e} \in \mathsf{E}^*$ as in $\mathsf{fp}_{\mathbf{e}} \stackrel{\text{def}}{=} \sum_{e \in \mathbf{e}} \tau_e$. Because each edge $v \to v' \in \mathsf{E}$ corresponds to the transition that moves a token from $(q_{\top}^{\lambda(v)}, v)$ to $(q_{\bot}^{\lambda(v)}, v)$ (*resp.* from $(q_{\bot}^{\lambda(v')}, v')$ to $(q_{\top}^{\lambda(v')}, v')$) in the PN $\beta(\mathsf{S})$, we shall abuse notation and write $\beta(v \to v')$ for the transition corresponding to $v \to v'$ in $\beta(\mathsf{S})$ and $\beta(\mathbf{e})$ for the sequence of transitions corresponding to $\mathbf{e} \in \mathsf{E}^*$.

We remark that, for each firing sequence $\mathsf{m} \stackrel{\beta(\mathbf{e})}{\rightsquigarrow} \mathsf{m}'$, we have $\mathsf{fp}_{\mathsf{m}'} - \mathsf{fp}_{\mathsf{m}} = \mathsf{fp}_{\mathbf{e}}$. Intuitively, $\mathsf{fp}_{\mathbf{e}}$ is a witness of the fact that the effect of firing the transitions $\beta(\mathbf{e})$ is to move

pebbles from $\{v \mid \mathsf{fp}_{\mathbf{e}}(v) = -1\}$ to $\{v \mid \mathsf{fp}_{\mathbf{e}}(v) = 1\}$; the vertices from $\{v \mid \mathsf{fp}_{\mathbf{e}}(v) = 0\}$ may store pebbles in between, but are ultimately restored to their initial state.

We denote by $\#_e(\mathbf{e})$ the number of times the edge e occurs in the sequence $\mathbf{e}$. We use the shorthands $\#_{u\to}(\mathbf{e}) \stackrel{\text{def}}{=} \sum_{u' \in \mathsf{V}} \#_{u\to u'}(\mathbf{e})$ and $\#_{\to u}(\mathbf{e}) \stackrel{\text{def}}{=} \sum_{u' \in \mathsf{V}} \#_{u'\to u}(\mathbf{e})$. We define the following partial orders between sequences of edges: $\mathbf{e}' \preceq \mathbf{e} \stackrel{\text{def}}{\iff} \#_e(\mathbf{e}') \leq \#_e(\mathbf{e})$, for each $e \in \mathsf{E}$, and $\mathbf{e}' \sqsubseteq \mathbf{e} \stackrel{\text{def}}{\iff} \mathbf{e}' \preceq \mathbf{e}$ and $\mathsf{fp}_{\mathbf{e}'} = \mathsf{fp}_{\mathbf{e}}$. The following lemma characterizes the existence of fireable sub-sequences:

Lemma 5. *For each marking* m *and sequence of edges* $\mathbf{e}$*, the following are equivalent:*
(i) $\mathsf{fp}_{\mathsf{m}} + \mathsf{fp}_{\mathbf{e}}$ *is a valid marking footprint,*
(ii) there exists a sequence of edges $\mathbf{e}' \sqsubseteq \mathbf{e}$ *such that* $\beta(\mathbf{e}')$ *is fireable from* m.

Proof. $(ii) \Rightarrow (i)$ is easy: if $\beta(\mathbf{e}')$ is fireable from m, then there exists some $\mathsf{m} \stackrel{\beta(\mathbf{e}')}{\rightsquigarrow} \mathsf{m}'$. We write $\mathsf{fp}_{\mathsf{m}'} - \mathsf{fp}_{\mathsf{m}} = \mathsf{fp}_{\mathbf{e}'}$, and the knowledge that $\mathsf{fp}_{\mathbf{e}'} = \mathsf{fp}_{\mathbf{e}}$ gives $\mathsf{fp}_{\mathsf{m}} + \mathsf{fp}_{\mathbf{e}} = \mathsf{fp}_{\mathsf{m}'}$ which is a marking footprint because m' is an automata-like marking.

For $(i) \Rightarrow (ii)$, we proceed by induction on the length of $\mathbf{e}$. When $\mathbf{e}$ is empty, an obvious (and in fact the only) candidate for $\mathbf{e}'$ is the empty sequence which has signature $\mathsf{fp}_{\varepsilon} = 0$ everywhere, and is obviously fireable.

In general when $\mathbf{e}$ is nonempty, we reinterpret $\mathbf{e}$ as a directed multigraph, where each edge $u \to v$ has multiplicity $\#_{(u\to v)}(\mathbf{e})$. As does any nonempty directed multigraph, either it contains an elementary cycle, or it is acyclic and it contains a maximal nonempty path.

- In the case of a cycle $\mathbf{e} \supseteq \mathbf{c} = u_0 \to u_1 \to u_2 \cdots u_{k-1} \to u_0$, we make the observation that $\mathsf{fp}_{\mathbf{c}} = 0$ everywhere. Note that an elementary cycle is characterized by every vertex involved having incoming and outgoing degree exactly 1 each, so $\mathsf{fp}_{\mathbf{c}}(u_i) = \mathsf{fp}_{\to u_i}(u_i) + \mathsf{fp}_{u_i \to}(u_i) = 1 - 1 = 0$. We deduce that $\mathbf{e} \setminus \mathbf{c} \sqsubseteq \mathbf{e}$, where $\mathbf{e} \setminus \mathbf{c}$ is naturally the sequence obtained by removing from $\mathbf{e}$ an occurrence of each transition from $\mathbf{c}$. Applying the inductive hypothesis to $\mathbf{e} \setminus \mathbf{c}$ which is strictly shorter than $\mathbf{e}$ gives $\mathbf{e}'$ fireable and $\mathbf{e}' \sqsubseteq \mathbf{e} \setminus \mathbf{c} \sqsubseteq \mathbf{e}$ where we conclude by transitivity of $\sqsubseteq$.
- Otherwise we have a directed acyclic graph, and a maximal path $\mathbf{e} \supseteq \mathbf{p} = u_0 \to u_1 \to u_2 \cdots u_{k-1} \to u_k$ of that graph. First observe that a path from u_0 to u_k has the same footprint as the single edge $u_0 \to u_k$ (evaluating to -1 on u_0, 1 on u_k, and all intermediate vertices cancel out). Secondly we have assumed by (i) that $0 \leq \mathsf{fp}_{\mathsf{m}_0}(u_0) + \mathsf{fp}_{\mathbf{p}}(u_0)$ which means $\mathsf{fp}_{\mathsf{m}_0}(u_0) = 1$, and $\mathsf{fp}_{\mathsf{m}_0}(u_k) + \mathsf{fp}_{\mathbf{p}}(u_k) \leq 1$ which means $\mathsf{fp}_{\mathsf{m}_0}(u_k) = 0$. The discrete quantity $\mathsf{fp}_{\mathsf{m}_0}(u_i)$, going from 1 at $i = 0$ to 0 at $i = k$, must for some value i_0 in between satisfy simultaneously $\mathsf{fp}_{\mathsf{m}_0}(u_{i_0}) = 1$ and $\mathsf{fp}_{\mathsf{m}_0}(u_{i_0+1}) = 0$. Thus $e_{i_0} = u_{i_0} \to u_{i_0+1}$ occurs in $\mathbf{e}$ and $\beta(e_{i_0})$ is fireable. This yields m_1 a marking such that $\mathsf{fp}_{\mathsf{m}_1} = \mathsf{fp}_{\mathsf{m}} + \mathsf{fp}_{e_{i_0}}$, on which we apply the inductive hypothesis for $\mathbf{e} \setminus e_{i_0}$ (at this point we need to show that $\mathsf{fp}_{\mathsf{m}_1} + \mathsf{fp}_{\mathbf{e}\setminus e_{i_0}}$ is a marking footprint, this comes from $\mathsf{fp}_{\mathsf{m}_1} + \mathsf{fp}_{\mathbf{e}\setminus e_{i_0}} = (\mathsf{fp}_{\mathsf{m}} + \mathsf{fp}_{e_{i_0}}) + (\mathsf{fp}_{\mathbf{e}} - \mathsf{fp}_{e_{i_0}}) = \mathsf{fp}_{\mathsf{m}} + \mathsf{fp}_{\mathbf{e}}$). Having thus obtained $\mathbf{e}'' \sqsubseteq \mathbf{e} \setminus e_{i_0}$ fireable from m_1, we construct $\mathbf{e}' \stackrel{\text{def}}{=} e_{i_0}; \mathbf{e}'' \sqsubseteq \mathbf{e}$ which is fireable from m.

We thus conclude that $\mathsf{fp}_{\mathsf{m}} + \mathsf{fp}_{\mathbf{e}}$ is a valid marking signature if and only if there exists some equivalent subsequence of $\mathbf{e}$ that is actually fireable from m. □

Using the previous lemma, we prove that, in order to cover a given marking $\mathsf{m}_{\mathsf{tgt}}$, it suffices to consider only those firing sequences that cross each vertex a bounded number of times, where the bound is the size of the unary encoding of $\mathsf{m}_{\mathsf{tgt}}$. To that end, we define the *degree* of a sequence $\mathbf{e} \in \mathsf{E}^*$ as the maximum number of occurrences of one vertex in the sequence, *i.e.,* $\deg(\mathbf{e}) \stackrel{\text{def}}{=} \max\{\#_{u\to}(\mathbf{e}), \#_{\to u}(\mathbf{e}) \mid u \in \mathsf{V}\}$. From now on, the domain of $\mathsf{m}_{\mathsf{tgt}}$ will implicitly be the set $Q \subseteq \mathsf{Q}_{\mathcal{P}}$. To simplify the following statement, we say that $\mathsf{m} : \mathsf{Q}_{\mathsf{N}} \to \mathbb{N}$ *covers* $\mathsf{m}_{\mathsf{tgt}}$ iff $\sum_{(q,v)\in\mathsf{Q}_{\mathsf{N}}} \mathsf{m}(q,v) \geq \mathsf{m}_{\mathsf{tgt}}(q)$, for each $q \in Q$ and that $\mathsf{m}_{\mathsf{tgt}}$ is *coverable* by S iff there exists $\mathsf{m} \in \mathrm{reach}(\beta(\mathsf{S}))$ that covers $\mathsf{m}_{\mathsf{tgt}}$. Since, in a pebble-passing system, markings can be equated to their footprints, we say that fp_{m} covers $\mathsf{m}_{\mathsf{tgt}}$ whenever m and $\mathsf{m}_{\mathsf{tgt}}$ satisfy the conditions (5) and (6) above.

Lemma 6. *A marking* $\mathsf{m}_{\mathsf{tgt}}$ *is coverable by* S *iff there exists* $\mathbf{e} \in \mathsf{E}^{*\leq K}$ *such that* $\mathsf{fp}_{\mathsf{m}_0} + \mathsf{fp}_{\mathbf{e}}$ *covers* $\mathsf{m}_{\mathsf{tgt}}$*, where* $K \stackrel{\text{def}}{=} \sum_{q\in Q} \mathsf{m}_{\mathsf{tgt}}(q)$*, and* $\mathsf{E}^{*\leq K} \stackrel{\text{def}}{=} \{\mathbf{e} \in \mathsf{E}^* \mid \deg(\mathbf{e}) \leq K\}$.

Proof. The ($\Leftarrow$) direction is easy, since the condition that $\mathsf{fp}_{\mathsf{m}_0} + \mathsf{fp}_{\mathbf{e}}$ covers $\mathsf{m}_{\mathsf{tgt}}$ alone implies that $\mathsf{m}_{\mathsf{tgt}}$ is coverable (through Lemma 5 and what it means for a footprint to cover a marking).

We focus on the proof ($\Rightarrow$), and assume that $\mathsf{m}_{\mathsf{tgt}}$ is coverable by S. In order to prove the existence of a sequence that satisfies the requirement, we proceed by as such: we assume that we have a minimal (in terms of length) firing sequence $\beta(\mathbf{e})$ that covers $\mathsf{m}_{\mathsf{tgt}}$ in S, deduce independently that it must satisfy be such that $\mathsf{fp}_{\mathsf{m}_0} + \mathsf{fp}_{\mathbf{e}}$ covers $\mathsf{m}_{\mathsf{tgt}}$ and assume by contradiction that this firing sequence must have degree at least $K+1$, in other words it must fire at least $K+1$ times an incoming or outgoing (assume without loss of generality that it is incoming, the other proof is identical) transition for some vertex u. We then construct a strictly shorter sequence that also covers the marking, thereby contradicting the hypothesis of minimality.

We first ensure that $\mathbf{e}$ is acyclic: a cycle $\mathbf{c}$ is characterized by $\mathsf{fp}_{\mathbf{c}} = 0$ everywhere, since in a cycle every vertex involved has as many incoming as outgoing edges occurring. Thus $\mathsf{fp}_{\mathbf{e}\setminus\mathbf{c}} = \mathsf{fp}_{\mathbf{e}} - \mathsf{fp}_{\mathbf{c}} = \mathsf{fp}_{\mathbf{e}}$: if $\mathsf{fp}_{\mathsf{m}_0} + \mathsf{fp}_{\mathbf{e}}$ is a marking signature, then so is $\mathsf{fp}_{\mathsf{m}_0} + \mathsf{fp}_{\mathbf{e}\setminus\mathbf{c}}$. Necessarily if $\beta(\mathbf{e})$ is fireable and $\mathbf{e}$ contains a cycle $\mathbf{c}$, then there is also an equivalent subsequence of $\mathbf{e} \setminus \mathbf{c}$ that is fireable. This would contradict the minimality of $\mathbf{e}$.

Now that $\mathbf{e}$ is acyclic, if there still exists a vertex u with in-degree at least $K+1$, denote by $i_1 < \cdots < i_{K+1}$ a set of indices of $K+1$ occurrences in $\mathbf{e}$ of an edge leading to u. In that situation, Figure 12 will compute a partial assignment p of edges to disjoint paths from 1 to $K+1$, such that all of these paths go through u

We thus have $K+1$ paths. Though these paths are not completely ordered, they do have a specific shape that guarantees that their endpoints are distinct. We call *positive path* an ordered path $(u_0 \to u_1);(u_1 \to u_2);(u_2 \to u_3);\ldots$ that occurs in that order as a subsequence of $\mathbf{e}$. We call *negative path* an ordered path $(u_1 \to u_0);(u_2 \to u_1);(u_3 \to u_2);\ldots$ that occurs in that order as a subsequence of $\mathbf{e}$. The paths we are interested in are interleavings of one positive path starting from u, and one negative path ending in u. For example, $(w_1 \to u);(u \to v_1);(w_2 \to w_1);(w_3 \to w_2);(v_1 \to v_2);(v_2 \to v_3);(w_4 \to w_3);(v_3 \to v_4);(v_4 \to v_5);(w_5 \to w_4)$ satisfies this criterion as it is an interleaving of the positive path $(u \to v_1);(v_1 \to v_2);(v_2 \to v_3);(v_3 \to v_4);(v_4 \to v_5)$ starting from u

input: $\mathbf{e} = (w_0 \to v_0);(w_1 \to v_1);\ldots$ a fireable sequence of edges (*i.e.,* $\beta(\mathbf{e})$ is a firing sequence)
input: u a vertex that occurs in $\mathbf{e}$
output: $p : [0, |\mathbf{e}|[\to \mathbb{N}^?$ a decomposition of $\mathbf{e}$ in paths with distinct endpoints

$n \leftarrow 0$
$p \leftarrow (\_ \mapsto \bot)$ {how to interpret p: if $p(i) = n$ it means that $\mathbf{e}_i$ is assigned to the n'th path}
for $|\mathbf{e}| > i \geq 0$ **decreasing do**
  invariant: every $p^{-1}(n')$ for $n' \leq n$ is a path
  if $v_i = u$ **then** {This edge is incoming for u, we want one path that goes through it}
    $p(i) \leftarrow n$
    $u^- \leftarrow w_i$
    $u^+ \leftarrow v_i$
    for $i < j < |\mathbf{e}|$ **increasing do**
      invariant: $p^{-1}(n)$ is a path from u^- to u^+
      if $p(j) = \bot$ **then** {$\mathbf{e}_i$ is not yet part of a path}
        if $w_j = u^+$ **then** {u^+ is the end of the path $p^{-1}(n)$, so $u^+ \to v_j$ can extend it}
          $p(j) \leftarrow n$
          $u^+ \leftarrow v_j$ {And of course v_j is the new end of $p^{-1}(n)$}
        end if
        if $v_j = u^-$ **then** {u^- is the start of the path $p^{-1}(n)$ so $w_j \to u^-$ can extend it}
          $p(j) \leftarrow n$
          $u^- \leftarrow w_j$ {And of conrse w_j is the new start of $p^{-1}(n)$}
        end if
      end if
    end for
    $n \leftarrow n+1$ {No other edges can be added to this path, so we handle the next}
  end if
end for

Fig. 12. Decomposing a sequence into paths. The fact that these paths have distinct endpoints is enforced by the specific shape that they have, as interleavings of one positive and one negative path that each independently appear in that order in $\mathbf{e}$.

and the negative path $(w_1 \to u);(w_2 \to w_1);(w_3 \to w_2);(w_4 \to w_3);(w_5 \to w_4)$ ending in u.

The reason these kinds of paths are relevant is that in the specific context where they occur as an ordered subsequence of a firing sequence of a pebble-passing system, the end of a maximal positive path has a pebble in the final marking, and the start of a maximal negative path has a hole in the final marking. Since it is impossible to move a pebble (*resp.* hole) to a vertex that already contains one, two maximal positive paths that use disjoint sets of edges cannot have the same end, and two maximal negative paths that use disjoint sets of edges cannot have the same start. By being interleavings of each a maximal positive path and a maximal negative path, the $K+1$ paths we have just created must have pairwise distinct starts and ends.

Put into equations this means each path $\mathbf{p}_j$ from u_0^j to $u_{k_j}^j$ contains the edge e_{i_j}, whose endpoints satisfy $u_0^j \neq u_0^{j'}$ and $u_{k_j}^j \neq u_{k_{j'}}^{j'}$ whenever $j \neq j'$, and because the paths are maximal we have $\mathsf{fp}_{\mathsf{m}'}(u_0^j) = 0$ and $\mathsf{fp}_{\mathsf{m}'}(u_{k_j}^j) = 1$ for every j. This last point implies

that each sequence $\mathbf{e} \setminus \mathbf{p_j}$ is fireable from m_0: $\mathsf{fp}_{\mathbf{p}_j}$ evaluates to -1 on u_0^j, 1 on $u_{k_j}^j$, and 0 everywhere else, and thus $0 \leq \mathsf{fp}_{\mathsf{m}'} - \mathsf{fp}_{\mathbf{p}_j} \leq 1$.

A coverability query, as we recall from Equations (5) and (6), is simply a requirement of cardinality for sets of the form $\{v \mid \mathsf{fp}_{\mathsf{m}'}(v) = 1\}$ or $\{v \mid \mathsf{fp}_{\mathsf{m}'}(v) = 0\}$ for specific process types. A marking $\mathsf{m}_{\mathsf{tgt}}$ can be covered by looking at K such vertices. The previous construction has given us $K+1$ pairs of one vertex of each of these two sets, so there is at least one path where both endpoints are irrelevant to the marking, in other words one path p_{j_0} from u_0 to u_k where in fact the inequalities $\|\{v \sim_{\ker(\lambda)} u_0 \mid (\mathsf{fp}_{\mathsf{m}_0} + \mathsf{fp}_{\mathbf{e}})(v) = 0\}\| > \mathsf{m}_{\mathsf{tgt}}(q_\perp^{\lambda(u_0)})$ and $\|\{v \sim_{\ker(\lambda)} u_k \mid (\pi_{\mathsf{m}_0} + \mathsf{fp}_{\mathbf{e}})(v) = 1\}\| > \mathsf{m}_{\mathsf{tgt}}(q_\top^{\lambda(u_k)})$ are strict. It follows that a final marking with one fewer pebble in $\lambda(u_k)$ and one fewer hole in $\lambda(u_0)$ would still cover $\mathsf{m}_{\mathsf{tgt}}$. One such marking is given by the application of Lemma 5 to $\mathbf{e} \setminus \mathbf{p}_{j_0}$ as mentioned above. We have thus reached a different marking than m' but that nevertheless still covers $\mathsf{m}_{\mathsf{tgt}}$, and done so in strictly fewer steps. This is a contradiction.

Thus the proof by contradiction ends and we deduce that there must exist a sequence that in addition to covering $\mathsf{m}_{\mathsf{tgt}}$ also has degree at most K.

4.3 Flows

To check coverability, we consider an algebra $\mathcal{F}$ whose elements represent the sequences of edges that cross each vertex at most K times. The domain of $\mathcal{F}$ is $\mathbb{F} \subseteq \mathrm{pow}([0,K]^\Sigma \times [0,K]^\Sigma \times [0,K]^Q)$. The elements $(f^+, f^-, n) \in \mathbb{F}$ represent sequences of edges, such that $f^+(\sigma)$ (*resp.* $f^-(\sigma)$) is the in-degree (*resp.* out-degree) of the σ-source of S and $n(q)$ is the number of tokens that end in $q \in Q$. Intuitively, a tuple (f^+, f^-, n) witnesses the existence of a sequence that covers n, that can later be combined with other sequences which compensate its surplus f^+ and deficit f^- on the sources of S. Since we consider only systems with finitely many sources (guaranteed by HR) and since $Q \subseteq \mathsf{Q}$ is finite, $\mathcal{F}$ is a finite algebra. We will later characterize its exact size. Formally, we define the following mappings, where $\mathbb{S}$ denotes the set of open systems, *i.e.,* systems with sources:

$$\omega : \mathsf{E}^{*\leq K} \times \mathsf{V}^\Sigma \to [0,K]^\Sigma \times [0,K]^\Sigma \times [0,K]^Q$$

$$\omega(\mathbf{e}, \xi) = (f^+, f^-, n) \overset{\mathsf{def}}{\iff} \begin{cases} f^+(\sigma) = \#_{\xi(\sigma)\to}(\mathbf{e}) \qquad f^-(\sigma) = \#_{\to\xi(\sigma)}(\mathbf{e}) \\ n(q_\perp^{\mathsf{p}}) = \min(\mathsf{m}_{\mathsf{tgt}}(q_\perp^{\mathsf{p}}), \|\{v \in \lambda^{-1}(\mathsf{p}) \setminus \mathrm{img}(\xi) \mid (\mathsf{fp}_{\mathsf{init}_{\mathsf{p}}} + \mathsf{fp}_{\mathbf{e}})(v) = 0\}\|) \\ n(q_\top^{\mathsf{p}}) = \min(\mathsf{m}_{\mathsf{tgt}}(q_\top^{\mathsf{p}}), \|\{v \in \lambda^{-1}(\mathsf{p}) \setminus \mathrm{img}(\xi) \mid (\mathsf{fp}_{\mathsf{init}_{\mathsf{p}}} + \mathsf{fp}_{\mathbf{e}})(v) = 1\}\|) \end{cases}$$

$$\eta : \mathbb{S} \to \mathrm{pow}([0,K]^\Sigma \times [0,K]^\Sigma \times [0,K]^Q)$$

$$\eta(\mathsf{S}, \xi) \overset{\mathsf{def}}{=} \{\omega(\mathbf{e}, \xi) \mid \mathbf{e} \in \mathsf{E}_{\mathsf{S}}^{*\leq K},\ \mathsf{fp}_{\mathsf{init}_{\beta(\mathsf{S})}} + \mathsf{fp}_{\mathbf{e}} \text{ is a valid marking footprint over } \mathsf{V}_{\mathsf{S}} \setminus \mathrm{img}(\xi)\}$$

We define the finite algebra of *flows* $\mathcal{F}$ using Proposition 1, where η is taken to be the homomorphism between $\mathcal{S}$ and $\mathcal{F}$:

Lemma 7. $\sim_{\ker(\eta)}$ *is a* HR *congruence.*

Proof. Excluding the trivial case of an edge, we perform a case analysis.

- $\underline{\mathsf{rename}^{\mathcal{S}}_{\alpha}}$: we assume $(\mathsf{S},\xi) \sim_{\ker(\eta)} (\mathsf{S}',\xi')$. We pick any $\omega(\mathbf{e},\xi\circ\alpha^{-1})$ in $\eta(\mathsf{rename}^{\mathcal{S}}_{\alpha}(\mathsf{S},\xi))$, and prove that it also occurs in $\eta(\mathsf{rename}^{\mathcal{S}}_{\alpha}(\mathsf{S}',\xi'))$. First we notice that $\mathsf{rename}^{\mathcal{S}}_{\alpha}(\mathsf{S},\xi)$ has the same edges as (S,ξ), and thus the same set of sequences of transitions. This means that $\omega(\mathbf{e},\xi) \in \eta(\mathsf{S},\xi)$. The initial equivalence gives the existence of a matching $\omega(\mathbf{e},\xi) = \omega(\mathbf{e}',\xi') \in \eta(\mathsf{S}',\xi')$, and thus a candidate is $\omega(\mathbf{e}',\xi'\circ\alpha^{-1})$. Since $\mathrm{img}(\xi)$ and $\mathrm{img}(\xi')$ are unchanged by a composition on the right by α^{-1}, we maintain $\omega(\mathbf{e}',\xi'\circ\alpha^{-1}) = \omega(\mathbf{e},\xi\circ\alpha^{-1})$ and thus $\eta(\mathsf{rename}^{\mathcal{S}}_{\alpha}(\mathsf{S},\xi)) \subseteq \eta(\mathsf{rename}^{\mathcal{S}}_{\alpha}(\mathsf{S}',\xi'))$. The symmetry of the definition makes this actually an equality.
- $\underline{\mathsf{restrict}^{\mathcal{S}}_{\tau}}$: we assume $(\mathsf{S},\xi) \sim_{\ker(\eta)} (\mathsf{S}',\xi')$. Once again $\mathsf{restrict}^{\mathcal{S}}_{\tau}(\mathsf{S},\xi)$ has the same edges as (S,ξ), so this time the difficulty comes from the fact that $\mathrm{img}(\xi)$ shrinks, not from the set of sequences to consider. The key observation to make is that we always have $\mathsf{fp}_{\mathbf{e}}(v) = \#_{\to v}(\mathbf{e}) - \#_{v\to}(\mathbf{e})$, and this holds in particular for $v = \xi(\sigma)$, where additionally $\mathsf{fp}_{\mathbf{e}}(\xi(\sigma)) = f^+(\sigma) - f^-(\sigma)$. What this means is that when the equivalence $\sim_{\ker(\eta)}$ imposes $f^+ = f'^+$ and $f^- = f'^-$, it also guarantees that $\mathsf{fp}_{\mathsf{init}_{\beta(\mathsf{S})}} + \mathsf{fp}_{\mathbf{e}}$ is a valid marking footprint on $\xi(\sigma)$ if and only if $\mathsf{fp}_{\mathsf{init}_{\beta(\mathsf{S}')}} + \mathsf{fp}_{\mathbf{e}'}$ is a valid marking footprint on $\xi'(\sigma)$. The same reasoning goes for the more specific criterion that $(\mathsf{fp}_{\mathsf{init}_{\mathsf{p}}} + \mathsf{fp}_{\mathbf{e}})(\xi(v)) = 0$ or 1, from which we get that after restriction the two tuples are still in accordance and thus $\eta(\mathsf{restrict}^{\mathcal{S}}_{\tau}(\mathsf{S},\xi)) = \eta(\mathsf{restrict}^{\mathcal{S}}_{\tau}(\mathsf{S}',\xi'))$.
- $\underline{\oplus^{\mathcal{S}}}$: take $(\mathsf{S}_1,\xi_1) \sim_{\ker(\eta)} (\mathsf{S}'_1,\xi'_1)$ and $(\mathsf{S}_2,\xi_2) \sim_{\ker(\eta)} (\mathsf{S}'_2,\xi'_2)$. We write $(\mathsf{S},\xi) = (\mathsf{S}_1,\xi_1) \oplus^{\mathcal{S}} (\mathsf{S}_2,\xi_2)$, and similarly for (S',ξ'). We consider a sequence $\mathbf{e} \in \mathsf{E}_{\mathsf{S}}^{*\leq K}$ and aim to prove that one can find $\mathbf{e}' \in \mathsf{E}_{\mathsf{S}'}^{*\leq K}$ that results in the same tuple through ω. By definition of $\oplus^{\mathcal{S}}$ we have $\mathsf{E}_{\mathsf{S}} = \mathsf{E}_{\mathsf{S}_1} \uplus \mathsf{E}_{\mathsf{S}_2}$. We can thus partition $\mathbf{e}$ into two subsequences: $\mathbf{e}_i$ holds all edges from S_i ($i = 1,2$). To these two subsequences naturally correspond through the equivalence $\sim_{\ker(\eta)}$ two sequences of $\mathsf{E}_{\mathsf{S}'_1}$ and $\mathsf{E}_{\mathsf{S}'_2}$ respectively, which we write $\mathbf{e}'_1$ and $\mathbf{e}'_2$. Since $\mathbf{e}' \stackrel{\mathsf{def}}{=} \mathbf{e}'_1;\mathbf{e}'_2$ is a good candidate for our solution, we take a moment to write the relationships between the various tuples involved:
  - $f^+ = f_1^+ + f_2^+$, $f'^+ = f_1'^+ + f_2'^+$, $f^- = f_1^- + f_2^-$, and $f'^- = f_1'^- + f_2'^-$: these hold because they are defined from elementary $\#_e(\mathbf{e})$ which due to the partition of edges satisfy additive properties;
  - $n(q) = n_1(q) + n_2(q)$, and $n'(q) = n'_1(q) + n'_2(q)$: these hold because during a composition the sets of vertices that are not sources are disjoint in the result;
  - and naturally $\mathsf{fp}_{\mathbf{e}} = \mathsf{fp}_{\mathbf{e}_1} + \mathsf{fp}_{\mathbf{e}_2}$, and $\mathsf{fp}_{\mathbf{e}'} = \mathsf{fp}_{\mathbf{e}'_1} + \mathsf{fp}_{\mathbf{e}'_2}$ because once again the sets of edges are disjoint each time.

  In the output of a composition, the set $\mathsf{V}_{\mathsf{S}} \setminus \mathrm{img}(\xi)$ of vertices that are not sources is a disjoint union of the vertices that were not sources from the two graphs that were composed, thus the condition "is a valid marking footprint over $\mathsf{V}_{\mathsf{S}} \setminus \mathrm{img}(\xi)$" is easily preserved. Thus $\eta(\mathsf{S},\xi) = \eta(\mathsf{S}',\xi')$.

We conclude that $\sim_{\ker(\eta)}$ is a HR congruence. □

This implies $\eta(\mathcal{L}^{\mathcal{S}}(\Gamma)) = \mathcal{L}^{\mathcal{T}}(\Gamma)$, so all we need is a way of computing $\mathcal{L}^{\mathcal{T}}(\Gamma)$. The remainder of the proof for the upper bound from Theorem 4 thus relies on the fact

that the interpretation of the HR signature in $\mathcal{F}$ is effectively computable, which is the purpose of the proposition below. The size of the grammar Γ is the total number of occurrences of a nonterminal or function symbol in a rule from Γ, denoted as $|\Gamma|$.

Proposition 3. *The size of each element $f \in \mathbb{F}$ is $2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}$ and the function $\mathsf{op}^{\mathcal{F}}(f_1,\ldots,f_n)$ can be computed in time $2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}$, for each HR-function symbol op of arity $n \geq 0$ and all elements $f_1,\ldots,f_n \in \mathbb{F}$. Moreover, for each grammar Γ using source labels from Σ, the language $\mathcal{L}^{\mathcal{F}}(\Gamma)$ is computable in time $2^{|\Gamma|\cdot 2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}}$.*

Proof. We denote by $\mathsf{closed}^{\xi}_{\tau}(\mathsf{p})$ the set of source labels such that $\sigma \in \mathsf{closed}^{\xi}_{\tau}(\mathsf{p})$ iff $\sigma \in \mathrm{dom}(\xi)$ and $\sigma \notin \tau$ and the $\mathrm{ptype}(\sigma) = \mathsf{p}$. We define $\mathsf{fp}_{\mathrm{init}}(\mathsf{p}) \stackrel{\mathrm{def}}{=} \mathsf{m}_0(q^{\mathsf{p}}_{\top})$. The function δ_x outputs 1 on x and 0 everywhere else (Kronecker delta). For a function f with domain Σ, we denote by $f\downarrow_\tau$ the restriction of f to the source labels in $\tau \subseteq \Sigma$. The inference rules below define the operations of $\mathcal{F}$:

$$\frac{0 \leq k \leq K}{(k\cdot\delta_{\sigma_2}, k\cdot\delta_{\sigma_1}, 0) \in (\mathsf{send},\mathsf{recv})^{\mathcal{F}}_{\sigma_1,\sigma_2}}\ \text{Edge}$$

$$\frac{(f^+,f^-,n) \in F}{(f^+\circ\alpha^{-1}, f^-\circ\alpha^{-1}, n) \in \mathsf{rename}^{\mathcal{F}}_{\alpha}(F)}\ \text{Rename}$$

$$\frac{\begin{array}{cc}(f_1^+,f_1^-,n_1)\in F_1 & (f_2^+,f_2^-,n_2)\in F_2\\ f_1^+ + f_2^+ \leq K & f_1^- + f_2^- \leq K\end{array}}{(f_1^+ + f_2^+, f_1^- + f_2^-, \min(\mathsf{m}_{\mathsf{tgt}}, n_1+n_2)) \in F_1 \oplus^{\mathcal{F}} F_2}\ \text{Compose}$$

$$\frac{\begin{array}{c}(f^+,f^-,n)\in F\\ \forall\sigma\in\mathsf{closed}^{\xi}_{\tau}(\mathsf{p}).\ f^+(\sigma)-f^-(\sigma)+\mathsf{fp}_{\mathrm{init}(\mathrm{ptype}(\sigma))}\in\{0,1\}\\ \forall\mathsf{p}.\ d_n(q^{\mathsf{p}}_{\bot}) \stackrel{\mathrm{def}}{=} \|\{\sigma\in\mathsf{closed}^{\xi}_{\tau}(\mathsf{p}) \mid f^+(\sigma)-f^-(\sigma)+\pi_{init}(\mathsf{p})=0\}\|\\ \forall\mathsf{p}.\ d_n(q^{\mathsf{p}}_{\top}) \stackrel{\mathrm{def}}{=} \|\{\sigma\in\mathsf{closed}^{\xi}_{\tau}(\mathsf{p}) \mid f^+(\sigma)-f^-(\sigma)+\pi_{init}(\mathsf{p})=1\}\|\end{array}}{(f^+\downarrow_\tau, f^-\downarrow_\tau, \min(\mathsf{m}_{\mathsf{tgt}}, n+d_n)) \in \mathsf{restrict}^{\mathcal{F}}_{\tau}(F)}\ \text{Restrict}$$

First we prove that these inference rules are compatible with η defined earlier, *i.e.,* for any ground HR term θ, we have $\eta(\theta^{\mathcal{S}}) = \theta^{\mathcal{F}}$.

- $\underline{(\mathsf{send},\mathsf{recv})^{\mathcal{F}}_{\sigma_1,\sigma_2}}$: the firing sequences of $(\mathsf{send},\mathsf{recv})^{\mathcal{S}}_{\sigma_1,\sigma_2}$ are a single edge repeated arbitrarily many times. Of these the firing sequences that use each vertex at most K times are exactly the $\mathbf{e}_k$ of footprint $\mathsf{fp}_{\mathbf{e}_k}$ that equals $-k$ on $\xi(\sigma_1)$, $+k$ on $\xi(\sigma_2)$, and 0 everywhere else. The set $\mathsf{V}\setminus\mathrm{img}(\xi)$ is empty and thus n is 0 everywhere. This generates exactly the set that the rule Edge produces.
- $\underline{\mathsf{rename}^{\mathcal{F}}_{\alpha}}$: by applying the same renaming to f^+ and f^- as we do to the graph, we obviously preserve the relationship between the two.

– $\underline{\oplus^{\mathcal{F}}}$: without the constraints of finiteness, we would simply take $f_1^+ + f_2^+$, $f_1^- + f_2^-$, and $n_1 + n_2$. The additional constraints are translated exactly from their equivalent in the definition of η: we require $f_1^+ + f_2^+ \leq K$ and $f_1^- + f_2^- \leq K$ to conform to the rule that the sequence we consider must belong to $\mathsf{E}^{*\leq K}$ (recall that f^+ and f^- are incoming and outgoing degrees, so if one of them exceeds K then the overall sequence has degree greater than K), and apply $\min(\mathsf{m}_{\mathsf{tgt}}, \ldots)$ so that n remains bounded by $\mathsf{m}_{\mathsf{tgt}}$ just like how it is defined in ω. Since we apply exactly the same constraints to (f^+, f^-, n) here as we did in the original definition of ω, we obtain the same final set of tuples.
– $\underline{\mathsf{restrict}_\tau^{\mathcal{F}}}$: the key property is that $\mathsf{closed}_\tau^\xi(\mathsf{p}) = \mathrm{ptype}^{-1}(\mathsf{p}) \cap (\mathrm{dom}(\xi) \setminus (\mathrm{dom}(\xi\downarrow_\tau)))$. This is relevant because it implies $(\lambda^{-1}(\mathsf{p}) \setminus \mathrm{img}(\xi\downarrow_\tau)) = (\lambda^{-1}(\mathsf{p}) \setminus \mathrm{img}(\xi)) \uplus \xi(\mathsf{closed}_\tau^\xi(\mathsf{p}))$, in which the first two terms have the same shape as one that occurs in the definition of n for ω, and the third gives the definition of d_n in rule Restrict above. This means that given n and n' from tuples in $\eta(\mathsf{S}, \xi)$ and $\eta(\mathsf{restrict}_\tau^{\mathcal{S}}(\mathsf{S}, \xi))$, we indeed have $n' = n + d_n$. The constraint $0 \leq f^+(\sigma) - f^-(\sigma) + \mathsf{fp}_{\mathrm{init}_{\mathrm{ptype}(\sigma)}} \leq 1$ from the premiss of the Restrict rule enforces the requirement "$\mathsf{fp}_{\mathrm{init}_{\beta(\mathsf{S})}} + \mathsf{fp}_{\mathbf{e}}$ is a valid marking footprint" for all vertices that were added to $\mathsf{V}_\mathsf{S} \setminus \mathrm{img}(\xi)$ by the restriction. The rest of the definition follows the same principles as for the other cases: $f^+(\sigma) - f^-(\sigma)$ has already been justified to be a synonym for $\mathsf{fp}_{\mathbf{e}}(\xi(\sigma))$, f^+ and f^- must be constrained to their new domain, and $n + d_n$ must not exceed $\mathsf{m}_{\mathsf{tgt}}$.

Therefore $\mathcal{F}$ defined explicitly here is compatible with its implicit definition in Lemma 7.

Since each tuple in an element of $\mathbb{F}$ is of size

$$\|[0,K]^\Sigma \times [0,K]^\Sigma \times [0,K]^Q\| = (K+1)^{2\|\Sigma\|+2\|\mathcal{P}\|}$$

each element of $\mathbb{F}$ is of size $2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}$. From the definition of the inference rules, Edge takes constant time to apply, Rename and Restrict take linear time, and Compose takes quadratic time in the size of their arguments. Then, evaluating the interpretation of any HR function symbol takes $2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}$ time. Because the domain of the algebra $\mathcal{F}$ is finite, the language $\mathcal{L}^{\mathcal{F}}(\Gamma)$ can be computed by a finite iteration of the Kleene sequence $\overrightarrow{\emptyset}, [\![\Gamma]\!](\overrightarrow{\emptyset}), [\![\Gamma]\!](\overrightarrow{\emptyset}) \cup [\![\Gamma]\!]^2(\overrightarrow{\emptyset}), \ldots$, where $[\![\Gamma]\!]$ is the function that maps any valuation of the nonterminals in Γ to the sets obtained by applying the rules of Γ and $\overrightarrow{\emptyset}$ assigns the empty set to each such nonterminal. Since each element of $\mathcal{F}$ is of size $2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}$, there are at most $2^{2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}}$ such elements, hence the Kleene iteration takes at most as many steps to reach a fixpoint. Moreover, computing each step of the Kleene iteration takes $|\Gamma| \cdot 2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}$ time, hence the entire language can be computed in time $2^{|\Gamma|\cdot 2^{O((\|\Sigma\|+\|\mathcal{P}\|)\cdot\log K)}}$. □

Deciding whether $\mathsf{m}_{\mathsf{tgt}}$ is coverable by some instance $\mathsf{S} \in \mathcal{L}^{\mathcal{S}}(\Gamma)$, for a given HR grammar Γ, is done by checking $\mathsf{restrict}_\emptyset^{\mathcal{F}}(\mathcal{L}^{\mathcal{F}}(\Gamma)) \cap \{(0,0,\mathsf{m}_{\mathsf{tgt}})\} \stackrel{?}{=} \emptyset$. We apply $\mathsf{restrict}_\emptyset$ to $\mathcal{L}^{\mathcal{F}}(\Gamma)$ to ensure that the sequences of edges considered lead to valid marking footprints on every vertex of a system $(\mathsf{S}, \xi) \in \mathcal{L}^{\mathcal{S}}(\Gamma)$, including the sources from $\mathrm{img}(\xi)$, that were exempt from satisfying this condition (see the above definition of η). Computing $\mathsf{restrict}_\emptyset^{\mathcal{F}}(\mathcal{L}^{\mathcal{F}}(\Gamma))$ simply adds a linear factor compared to $\mathcal{L}^{\mathcal{F}}(\Gamma)$, and so does checking if it contains $(0,0,\mathsf{m}_{\mathsf{tgt}})$. Thus we have an overall 2EXPTIME upper bound.

Name	Architecture	PPS	TPS	Result	Size	Runtime (ms)
`fig-1c`	chain		✓	2/2	$(\bigcirc 12, \square 13) \times 2$	$(59+9) \pm 6$
`fig-1d`	star		✓	2/2	$(\bigcirc 11, \square 10) \times 2$	$(59+9) \pm 4$
`fig-5`	trivial	✓	✓	1/2	$(\bigcirc 13, \square 10) \times 3$	$(49+2) \pm 4$
`ring`	ring	✓	✓	2/2	$(\bigcirc 9, \square 11) \times 16$	$(80+21) \pm 4$
`double-ring`	ring	✓		1/1	$(\bigcirc 8, \square 14) \times 34$	$(113+44) \pm 10$
`philos`	ring			1/1	$(\bigcirc 16, \square 14) \times 8$	$(134+21) \pm 5$
`consensus`	star			5/5	$(\bigcirc 29, \square 23) \times 6$	$(83+26) \pm 6$
`leader-election`	ring			1/2	$(\bigcirc 27, \square 32) \times 2$	$(58+35) \pm 5$
`tree-dfs`	binary tree		✓	2/2	$(\bigcirc 19, \square 16) \times 2$	$(52+5) \pm 4$
`tree-down`	binary tree	✓	✓	1/1	$(\bigcirc 11, \square 12) \times 20$	$(125+36) \pm 7$
`tree-halves`	binary tree			4/4	$(\bigcirc 26, \square 23) \times 8$	$(92+190) \pm 7$
`tree-nav`	chained binary tree	✓	✓	2/2	$(\bigcirc 12, \square 19) \times 12$	$(139+46) \pm 7$
`lock`	star		✓	1/3	$(\bigcirc 9, \square 11) \times 4$	$(59+8) \pm 5$
`star`	star	✓	✓	3/3	$(\bigcirc 12, \square 11) \times 4$	$(58+11) \pm 5$
`star-ring`	chained star		✓	3/3	$(\bigcirc 17, \square 14) \times 2$	$(63+12) \pm 4$
`server-loop`	ring of stars			2/3	$(\bigcirc 19, \square 19) \times 8$	$(112+4627) \pm 20$
`coverapprox`	star			1/2	$(\bigcirc 3, \square 8) \times 6$	$(70+62) \pm 8$
`simplify-me`	star			1/2	$(\bigcirc 7, \square 9) \times 4$	$(64+21) \pm 5$
`propagation`	ring			1/2	$(\bigcirc 13, \square 13) \times 4$	$(90+278) \pm 10$
`open`	ring			0/1	$(\bigcirc 13, \square 14) \times 4$	$(74+269) \pm 13$

Fig. 13. Table of experiments. "PPS", these are pebble-passing systems (Section 4). "TPS", these are token-passing systems ([4]). "Result", S/T means that of T safety properties, we proved S. "Size", $(\bigcirc p, \square t) \times n$ means that LoLA analyzed n nets, consisting of on average p places and t transitions. "Runtime", $(p+l) \pm e$ means that ParCoSys ran for $p+l$ milliseconds with a standard deviation of e milliseconds, including p computing the abstraction and l waiting for LoLA.

The PSPACE lower bound is obtained by a polynomial reduction from the PSPACE-complete emptiness problem for 2-way nondeterministic finite automata (2NFA). The idea of the reduction is to simulate a run of a 2NFA on a given word by a grid-like system. This system encodes each letter of the word horizontally, and each state of the automaton vertically. The possible movements of the pebble are determined by the transition relation of the 2NFA. Since there are finitely many states in the automaton, we have a grid of constant height and unbounded width, which is expressible in HR.

5 Experiments

We have implemented the counting abstraction method in the prototype tool ParCoSys (**Par**ameterized **Co**verability) [30]. The input of the tool is a grammar describing the system and a safety property to be checked. The output is a finite set of Petri nets that is fed to the LoLA analyzer [32]. Our choice for LoLA was driven by its robustness and performance, but any Petri net analyzer can be used as back-end, in principle.
We tested[7] our tool on the examples listed in Figure 13

- `fig-1c` and `fig-1d` are the systems used as examples in Figure 1 (c) and (d) respectively. We easily verify nonduplication of the token ($\mathsf{m}_{\mathsf{tgt}}(tok) + \mathsf{m}_{\mathsf{tgt}}(tokC) > 1$) and mutual exclusion of processes in the critical section *work* ($\mathsf{m}_{\mathsf{tgt}}(work) > 1$).
- `fig-5` is the example shown in Figure 10. Here the "Result" 1/2 denotes that $\mathsf{m}_{\mathsf{tgt}}(q_c^1) > 0$ exhibits a false positive with the default folding, but there is an easy

[7] The times were obtained on a Intel Ultra 7 laptop, with 16GiB RAM, under Ubuntu 24.04. All benchmark specifications and logs are provided as additional material [30].

refinement of the grammar that leads to a successful verification on the second attempt.
- `ring` is a standard token ring, on which we verify mutual exclusion ($\mathsf{m}_{\mathsf{tgt}}(tok) > 1$).
- `double-ring` is a token ring with two tokens instead of one ($\mathsf{m}_{\mathsf{tgt}}(tok) > 2$). It is mainly interesting as an example of a PPS that is *not* TPS.
- `philos` implements the algorithm of dining philosophers. We prove that adjacent processes p_1 and p_2 do not enter their critical section *eating* simultaneously ($\mathsf{m}_{\mathsf{tgt}}(eating^{p_1}) > 0 \wedge \mathsf{m}_{\mathsf{tgt}}(eating^{p_2}) > 0$).
- `consensus` has a star of processes performing 2-valued consensus. All processes must choose the same value: ($\mathsf{m}_{\mathsf{tgt}}(choose_0) > 0 \wedge \mathsf{m}_{\mathsf{tgt}}(choose_1) > 0$)).
- `leader-election` performs a leader election with predetermined winner (found in [9]). We prove that p_0 is the unique winner ($\mathsf{m}_{\mathsf{tgt}}(win^{p_0}) > 0 \wedge \mathsf{m}_{\mathsf{tgt}}(win) > 0$).

The rest of the tests are more *ad hoc*, designed to showcase a wider range of architectures as well as interesting false positives:

- `tree-dfs`, `tree-down`, `tree-halves`, `tree-nav` are binary trees on which we prove different ways of expressing mutual exclusion with one or two tokens.
- `lock` should satisfy a mutual exclusion ($\mathsf{m}_{\mathsf{tgt}}(on) > 1$), but a false positive occurs. A partial unfolding (Section 3.4) allows proving the desired property.
- `star` and `star-ring` are stars with easy mutual exclusion properties.
- `server-loop` is a ring where each node has itself a star of child processes. One false positive here could theoretically be solved by a partial unfolding, but we have not yet implemented the logic that would allow for this specific pattern.
- `coverapprox`, `simplify-me` and `propagation` each have a false positive that can be solved by a refinement technique consisting of deleting transitions that are found to be unfireable during intermediate stages of the fixed point computation.
- `open` so far evades our ideas for refinements. We can prove that partial unfolding on its own is insufficient, since all finite foldings exhibit false positives.

6 Conclusions

We present two orthogonal verification results for parameterized process networks with topology specified using hyperedge-replacement graph grammars. The first result is a finitary counting abstraction, that consists in collapsing nodes of the same type in the parameterized family of Petri nets that gives the semantics of behaviors. The second result identifies a decidable fragment of the (undecidable) parameterized verification problem and evaluates its complexity bounds.

Acknowledgements. This research is supported by the French National Research Agency project "Parametic Verification of Dynamic Distributed Systems" (PaVeDyS) under grant number ANR-23-CE48-0005.

Disclosure of interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. P. A. Abdulla, K. Cerans, B. Jonsson, and Y. Tsay. General decidability theorems for infinite-state systems. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 313–321. IEEE Computer Society, 1996.
2. P. A. Abdulla, G. Delzanno, N. B. Henda, and A. Rezine. Monotonic abstraction: on efficient verification of parameterized systems. *Int. J. Found. Comput. Sci.*, 20(5):779–801, 2009.
3. P. A. Abdulla, G. Delzanno, and A. Rezine. Approximated parameterized verification of infinite-state processes with global conditions. *Formal Methods Syst. Des.*, 34(2):126–156, 2009.
4. B. Aminof, S. Jacobs, A. Khalimov, and S. Rubin. Parameterized model checking of token-passing systems. In *VMCAI'14*, volume 8318 of *Lecture Notes in Computer Science*, pages 262–281. Springer, 2014.
5. B. Aminof, T. Kotek, S. Rubin, F. Spegni, and H. Veith. Parameterized model checking of rendezvous systems. *Distributed Comput.*, 31(3):187–222, 2018.
6. B. Aminof and S. Rubin. Model checking parameterised multi-token systems via the composition method. In *IJCAR'16*, volume 9706 of *Lecture Notes in Computer Science*, pages 499–515. Springer, 2016.
7. R. Bloem, S. Jacobs, and A. Khalimov. *Decidability of Parameterized Verification*. Morgan & Claypool Publishers, 2015.
8. R. Bloem, S. Jacobs, A. Khalimov, I. Konnov, S. Rubin, H. Veith, and J. Widder. *Decidability of Parameterized Verification*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool Publishers, 2015.
9. M. Bozga, J. Esparza, R. Iosif, J. Sifakis, and C. Welzel. Structural invariants for the verification of systems with parameterized architectures. In A. Biere and D. Parker, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 26th International Conference, TACAS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings, Part I*, volume 12078 of *Lecture Notes in Computer Science*, pages 228–246. Springer, 2020.
10. M. Bozga, R. Iosif, A. Sangnier, and N. Villani. Counting abstraction for the verification of structured parameterized networks (full version), 2025. arxiv.org/abs/2502.15391.
11. M. Bozga, R. Iosif, and J. Sifakis. Checking deadlock-freedom of parametric component-based systems. *J. Log. Algebraic Methods Program.*, 119:100621, 2021.
12. M. Bozga, R. Iosif, and J. Sifakis. Verification of component-based systems with recursive architectures. *Theor. Comput. Sci.*, 940(Part):146–175, 2023.
13. J. Bradbury, J. Cordy, J. Dingel, and M. Wermelinger. A survey of self-management in dynamic software architecture specifications. In *Proceedings of the 1st ACM SIGSOFT workshop on Self-managed systems*, pages 28–33. ACM, 2004.
14. M. Browne, E. Clarke, and O. Grumberg. Reasoning about networks with many identical finite state processes. *Information and Computation*, 81(1):13 – 31, 1989.
15. A. Butting, R. Heim, O. Kautz, J. O. Ringert, B. Rumpe, and A. Wortmann. A classification of dynamic reconfiguration in component and connector architecture description. In *Proceedings of MODELS 2017 Satellite Event: Workshops (ModComp)*, volume 2019 of *CEUR Workshop Proceedings*, pages 10–16. CEUR-WS.org, 2017.
16. Y. Chen, C. Hong, A. W. Lin, and P. Rümmer. Learning to prove safety over parameterised concurrent systems. In D. Stewart and G. Weissenbacher, editors, *2017 Formal Methods in Computer Aided Design, FMCAD 2017*, pages 76–83. IEEE, 2017.
17. B. Courcelle and J. Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 2012.

18. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. In *Fundamentals of Computation Theory*, pages 221–232. Springer Berlin Heidelberg, 1995.
19. J. Esparza, M. A. Raskin, and C. Welzel. Regular model checking upside-down: An invariant-based approach. In B. Klin, S. Lasota, and A. Muscholl, editors, *33rd International Conference on Concurrency Theory, CONCUR 2022, September 12-16, 2022, Warsaw, Poland*, volume 243 of *LIPIcs*, pages 23:1–23:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
20. A. Finkel and J. Leroux. Recent and simple algorithms for petri nets. *Softw. Syst. Model.*, 14(2):719–725, 2015.
21. Z. Galil. Hierarchies of complete problems. *Acta Informatica*, 1976.
22. S. M. German and A. P. Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992.
23. D. Hirsch, P. Inverardi, and U. Montanari. Graph grammars and constraint solving for software architecture styles. In *Proceedings of the Third International Workshop on Software Architecture*, ISAW '98, page 69–72, New York, NY, USA, 1998. Association for Computing Machinery.
24. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256(1):93 – 112, 2001.
25. Y. Kesten, A. Pnueli, E. Shahar, and L. D. Zuck. Network invariants in action. In *CONCUR 2002 - Concurrency Theory, 13th International Conference*, volume 2421 of *LNCS*, pages 101–115. Springer, 2002.
26. D. Le Metayer. Describing software architecture styles using graph grammars. *IEEE Transactions on Software Engineering*, 24(7):521–533, 1998.
27. D. Lesens, N. Halbwachs, and P. Raymond. Automatic verification of parameterized linear networks of processes. In *The 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 346–357. ACM Press, 1997.
28. A. W. Lin and P. Rümmer. Regular model checking revisited. In E. Olderog, B. Steffen, and W. Yi, editors, *Model Checking, Synthesis, and Learning - Essays Dedicated to Bengt Jonsson on The Occasion of His 60th Birthday*, volume 13030 of *Lecture Notes in Computer Science*, pages 97–114. Springer, 2021.
29. Z. Shtadler and O. Grumberg. Network grammars, communication behaviors and automatic verification. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems, International Workshop*, volume 407 of *LNCS*, pages 151–165. Springer, 1989.
30. N. Villani, R. Iosif, A. Sangnier, and M. Bozga. Parcosys: Counting abstraction for the verification of structured parameterized networks, Apr. 2025. Ongoing development version at `https://gricad-gitlab.univ-grenoble-alpes.fr/neven/parcosys`.
31. K. Wolf. Petri net model checking with lola 2. In V. Khomenko and O. H. Roux, editors, *Application and Theory of Petri Nets and Concurrency - 39th International Conference, PETRI NETS 2018, Bratislava, Slovakia, June 24-29, 2018, Proceedings*, volume 10877 of *Lecture Notes in Computer Science*, pages 351–362. Springer, 2018.
32. K. Wolf and N. Lohmann. Lola : A low level petri net analyzer. `https://theo.informatik.uni-rostock.de/theo-forschung/tools/lola/`.
33. P. Wolper and V. Lovinfosse. Verifying properties of large sets of processes with network invariants. In *Automatic Verification Methods for Finite State Systems, International Workshop*, volume 407 of *LNCS*, pages 68–80. Springer, 1989.