Verifying Parameterized Networks Specified by Vertex-Replacement Graph Grammars

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 parametric reachability problem (PRP) for families of networks described by vertex-replacement (VR) graph grammars, where network nodes run replicas of finite-state processes that communicate via binary handshaking. We show that the PRP problem for VR grammars can be effectively reduced to the PRP problem for hyperedge-replacement (HR) grammars, at the cost of introducing extra edges for routing messages. This transformation is motivated by the existence of several parametric verification techniques for families of networks specified by HR grammars, or similar inductive formalisms. Our reduction enables applying the verification techniques for HR systems to systems with dense architectures, such as user-specified cliques and multi-partite graphs.

1 Introduction

As pointed out in recent literature, "*network verification is a necessary part of deploying modern hyperscale datacenters*" [19]. In this context, verification considers global properties of routing control, such as access between point A and point B. Very often, the intended functionality of a network can be derived from its architecture, typically known at early stages of network design.

The huge scale of present-day datacenters, of $\sim 10^4$ routers for a regional hub and continuously growing, requires *parametric* verification techniques, that work no matter how many servers in a rack, switches in a cluster, clusters in a layer, etc. Despite decades of theoretical research and an impressive body of results (see [7] for a nice survey), these techniques consider mostly hard-coded architectures, either dense (*e.g.,* cliques [16]) or sparse (*e.g.,* rings [11]) families of graphs with a common topological pattern.

The problem of verifying networks with user-specified architecture has gained recent momentum, with the development of parametric verification techniques for systems described using logic [5] or graph grammars [20]. In principle, logic-based graph specification languages are good at describing global properties, such as planarity, k-colorability or reachability, which make them more suitable for the specification of correctness properties, than network design.

On the other hand, graph grammars are appealing for their constructive aspect, *i.e.,* defining how large graphs are built from smaller subgraphs. Moreover, this recursive way of describing sets of graphs is common among programmers and software engineers, who are familiar with inductively defined datastructures (lists, trees, etc.)

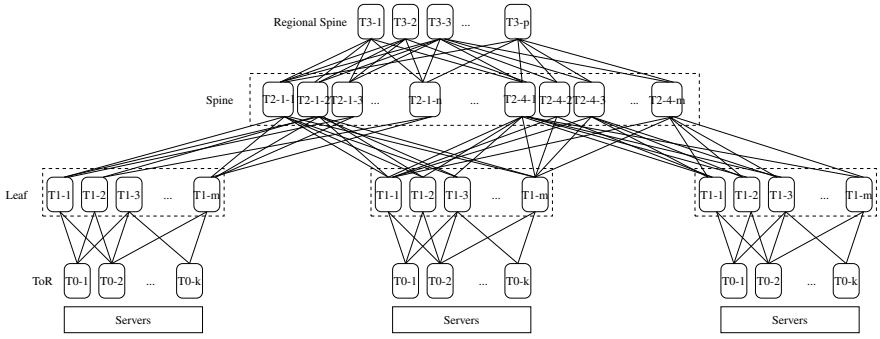


Fig. 1. Azure Datacenter Switching Topology

used in both imperative and functional programming. Graph grammars are at the core of the theory of formal languages (see [13] for a comprehensive survey). Based on the underlying set of operations (*i.e.,* graph algebra), we distinguish between *vertex-replacement* (VR) and *hyperedge-replacement* (HR) grammars. 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. On the other hand, *vertex-replacement* (VR) grammars are strictly more expressive [12]. For instance, VR grammars can describe the sets $\{K_n\}_{n\geq 1}$ of cliques and $\{K_{n,m}\}_{n,m\geq 1}$ of complete bipartite graphs, among others.

A concrete motivation for using VR grammars to design datacenter networks can be found in the informal description of the Azure datacenter architecture [17,19], deployed in the Microsoft hyperscale cloud network (Figure 1). The *top-of-rack* (ToR) switches connect servers hosted in a rack. Several ToR switches are connected together by a number of *leaf* switches. Leaf switches are in turn connected together by a set of *spine* switches, that connect the datacenter to the Azure regional network. These networks cannot be described as cliques, because of their layered structure. However, the links between each layer (*i.e.,* ToR, leaf, spine, regional spine) and its adjacent layers form a complete bipartite graph, thus lying outside the scope of HR grammars, which cannot define complete bipartite graphs of unbounded sizes[3].

The precise relation between the expressivity of VR and HR grammars has been elegantly coined in a result by Courcelle, stating that a set defined by a VR grammar can also be defined by an HR grammar if and only if it is sparse [12]. Moreover, each VR grammar can be transformed into an HR grammar producing a set of graphs with ε-edges, such that the elimination of these edges from the HR language yields the original VR language. For instance, the elimination of ε-edges from the graph from Figure 2 (b) yields the complete bipartite graph $K_{4,3}$ from Figure 2 (a).

Contribution We leverage the idea of transforming VR grammars into HR grammars modulo elimination of ε-edges to define a reduction from the parametric reachability

[3] Each set $\{K_{n,m}\}_{n\in N,m\in M}$ for N,M infinite subsets of $\mathbb{N}$, has unbounded tree-width, whereas HR-grammars define bounded tree-width sets.

Fig. 2. The complete bipartite graph $\overrightarrow{K}_{4,3}$ (a) and one possible encoding using ε-edges (b).

problem (PRP) for systems described by VR grammars and finite-state local behaviors, with correctness properties given by unrestricted arithmetic formulæ over a set of variables, tracking how many processes are in a given local state, to the PRP problem for systems described by HR grammars, having the same correctness property. This is motivated by several recent developments in the verification of parametric systems with HR-style architecture descriptions. In particular, [10] reports on an inference of structural invariants (*i.e.,* over-approximations of the reachable set, that can be derived from the structure of the network) for architectures described using a variant of Separation Logic with inductive definitions (same expressivity as HR grammars) Moreover, [9] gives an abstraction technique that folds similar network nodes into a finite Petri net, whose coverability problem over-approximates the original parametric coverability problem. In this paper, we give a more general reduction method, that preserves all safety properties specified by arithmetic assertions, rather than the particular case of coverability.

Related Work The literature on parametric verification is too large to be recalled here. We point to [7] for a survey on the (un)decidability results, typically concerning clique or ring architectures. The specification of network architectures using inductive definitions can be traced back to the work on *network grammars* [22,20,18], that use inductive rules to describe systems with linear (pipeline, token-ring) architectures. These initial works target mainly safety properties, by inferring network invariants [23,21]. Pipeline and clique architectures are also considered by methods based on the theory of well-structured transition systems [1], such as monotonic abstraction [2,3].

A prominent exception, that considers the verification of clique-width bounded architectures specified using Monadic Second Order Logic (MSO) against $\mathsf{CTL}^* \setminus \bigcirc$ properties is [5]. Their result uses a reduction from the parametric model checking problem (PMCP) to the decidability of MSO over graphs of bounded clique-width [13]. Our result is orthogonal, because we consider VR grammars that strictly subsume the MSO-definable sets of bounded clique-width. It is, however, an interesting question if our reduction can be used to establish a more general decidability result. A promising lead would be to use the decidability of PMCP for HR systems with local behaviors that mimick the passing of a pebble from one node to another [9].

Notation We denote by $\mathbb{Z}$ ($\mathbb{N}$) the set of (positive) integers. 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 written $\|A\|$. A singleton $\{a\}$ is simply denoted a. The union of two disjoint sets A and B is denoted as $A \uplus B$. The Cartesian product of two sets A and B is denoted $A \times B$. As usual, we define A^* the set of (possibly empty) finite sequences of elements from A, and A^∞ the set of finite or infinite sequences over A. Finally $\mathcal{P}(A)$ is the powerset of A.

2 Graphs

For simplicity reasons, in this paper we consider only networks whose topologies are described by oriented binary graphs, where vertices model processes and edges model a symmetric rendezvous between two processes, *i.e.,* we do not distinguish the process initiating the communication and consider that both participants have equal roles.

Let Λ and Δ be finite disjoint alphabets of vertex and edge labels, respectively. A *graph* is a tuple $\mathsf{G} = (\mathsf{V_G}, \mathsf{E_G}, \lambda_\mathsf{G})$, where $\mathsf{V_G}$ is a finite set of vertices, $\mathsf{E_G} \subseteq \mathsf{V_G} \times \Delta \times \mathsf{V_G}$ is a set of edges, such that $(v, \delta, v') \in \mathsf{E_G}$ implies $v \neq v'$ (*i.e.,* graphs have no self-loops) and $\lambda_\mathsf{G} : \mathsf{V_G} \to \mathcal{P}(\Lambda)$ assigns a set of labels to each vertex. We do not distinguish graphs that are isomorphic. We denote by $\mathcal{G}(\Lambda, \Delta)$ the set of graphs with vertex and edge labels from Λ and Δ, or simply $\mathcal{G}$, if Δ and Λ are understood.

We shall consider parameterized networks described as infinite sets of graphs. Albeit infinite, these sets have a finite description, given by finitely many inductive rules stating how the graphs in the set are built from smaller graphs. This constructive[4] approach to the specification of infinite sets of graphs relies on graph algebras, being at the core of an impressive body of theoretical work (see [13] for a survey).

2.1 Algebras

We present two classical graph algebras, namely *vertex-replacement* (VR) and *hyperedge-replacement* (HR). Let Λ and Δ be fixed in the following. We consider a set $\Pi \subseteq \Lambda$ of distinguished vertex labels, called *ports*, such that $\|\lambda_\mathsf{G}(v) \cap \Pi\| \leq 1$, for each graph $\mathsf{G} \in \mathcal{G}(\Lambda, \Delta)$ and each vertex $v \in \mathsf{V_G}$. In other words, a vertex is labeled with at most one port. We say that a vertex $v \in \mathsf{V_G}$ is a π-port of G whenever $\pi \in \lambda_\mathsf{G}(v)$. The *sort* of a graph is the set of ports that occur in the labels of the vertices from the graph.

The labels from $\Lambda \setminus \Pi$ will be needed later to associate vertices with local behaviors, referred to as process types. For the time being, we assume the set of process types to be $\Lambda \setminus \Pi = \{\lambda_1, \ldots, \lambda_n\}$ and that the set of port labels is partitioned accordingly, *i.e.,* $\Pi = \Pi_{\lambda_1} \uplus \ldots \uplus \Pi_{\lambda_n}$. We say that the *process type* of a port $\pi \in \Pi$ is $\lambda \in \Lambda \setminus \Pi$, denoted $\mathsf{ptype}(\pi) = \lambda \stackrel{\mathsf{def}}{\iff} \pi \in \Pi_\lambda$. A partial function $\alpha : \Pi \rightharpoonup \Pi$ is *type-preserving* iff $\alpha(\pi)$, if defined, has the same process type as π, for each $\pi \in \Pi$.

The domain of the *VR-algebra* is the set $\mathcal{G}$ of graphs. The operations of VR are the following, see Figure 3 (a) for an illustration:

- *Disjoint union*: $\mathsf{G}_1 \oplus_{\mathbf{p}_1, \mathbf{p}_2} \mathsf{G}_2 \stackrel{\mathsf{def}}{=} H$ iff the sorts of G_1 and G_2 are $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively, $\mathsf{V}_H = \mathsf{V_{G_1}} \uplus \mathsf{V_{G_2}}$, $\mathsf{E}_H = \mathsf{E_{G_1}} \uplus \mathsf{E_{G_2}}$ and $\lambda_H = \lambda_{\mathsf{G}_1} \uplus \lambda_{\mathsf{G}_2}$ where, if $\mathsf{V_{G_1}} \cap \mathsf{V_{G_2}} \neq \emptyset$ or $\mathsf{E_{G_1}} \cap \mathsf{E_{G_2}} \neq \emptyset$, we replace G_2 by an isomorphic copy disjoint from G_1. We write $\oplus$ instead of $\oplus_{\mathbf{p}_1, \mathbf{p}_2}$ when the argument sorts are understood.
- *Edge creation*: for an edge label $\delta \in \Delta$ and port labels $\pi_1 \neq \pi_2 \in \Pi$, $\mathsf{add}_{\delta, \pi_1, \pi_2}(\mathsf{G}) \stackrel{\mathsf{def}}{=} H$ iff $\mathsf{V}_H = \mathsf{V_G}$, $\mathsf{E}_H = \mathsf{E_G} \cup \{(v_1, \delta, v_2) \mid v_1, v_2 \in \mathsf{V_G},\ \pi_i \in \lambda_\mathsf{G}(v_i),\ i = 1, 2\}$ and $\lambda_H = \lambda_\mathsf{G}$; in other words, $\mathsf{add}_{\delta, \pi_1, \pi_2}$ adds a directed δ-edge from each π_1-port to each π_2-port of its argument. No edge is added if such an edge already exists. Moreover, because π_1

[4] As opposed to the descriptive method of defining sets of graphs by their common properties, using *e.g.,* monadic second-order logic.

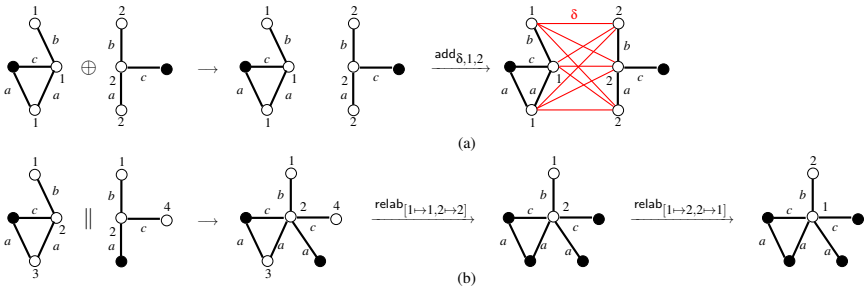


Fig. 3. VR operations (a) HR operations (b). Ports are depicted as shallow circles; port labels are natural numbers.

and π_2 are different labels and each vertex has at most one port label, no self-loops are introduced by this operation.

- *Port redefinition*: for a type-preserving partial function $\alpha : \Pi \rightharpoonup \Pi$, $\mathsf{relab}_\alpha(\mathsf{G}) \stackrel{\text{def}}{=} H$ iff $\mathsf{V}_H = \mathsf{V}_\mathsf{G}$, $\mathsf{E}_H = \mathsf{E}_\mathsf{G}$ and $\lambda_H(v) = \{\alpha(\pi) \mid \pi \in \lambda_\mathsf{G}(v) \cap \Pi\} \cup (\lambda_\mathsf{G}(v) \setminus \Pi)$, for each $v \in \mathsf{V}_H$, *i.e.,* the port labels are redefined according to α (if $\alpha(\pi)$ is undefined, the port label is erased) and the other labels are unchanged.
- *Single-vertex graphs*: for a port label $\pi \in \Pi$, $\bullet\pi \stackrel{\text{def}}{=} H$ iff $\mathsf{V}_H = \{u\}$ for some vertex u, $\mathsf{E}_H = \emptyset$ and $\lambda_H(u) = \{\pi, \mathrm{ptype}(\pi)\}$.

For simplicity, we use the same notation for a VR operation above and its corresponding function symbol. Since Δ and Π are finite sets, the above set of function symbols is finite. A VR-term $\theta[x_1, \ldots, x_n]$ consists of function symbols (*i.e.,* the binary $\oplus$ and unary $\mathsf{add}_{\delta,\pi_1,\pi_2}$ and relab_α) and the variables $x_1, \ldots, x_n$ of arity zero. A term is ground if it has no occurrences of variables. We write $\mathrm{val}^{\mathsf{VR}}(\theta)$ for the graph obtained by interpreting the function symbols in the ground term θ as the corresponding VR-operations.

Example 1. The VR-term $\theta \stackrel{\text{def}}{=} \mathsf{relab}_\emptyset(\mathsf{add}_{a,\pi,\pi'}((\bigoplus_{i\in 1,\ldots,4} \bullet\pi) \oplus (\bigoplus_{j\in 1,\ldots,3} \bullet\pi')))$ evaluates to the complete bipartite graph $\mathrm{val}^{\mathsf{VR}}(\theta) = \overrightarrow{K}_{4,3}$ with directed edges a, see Figure 2 (a). Edge labels are $\Delta = \{a\}$, port labels are $\Pi = \{\pi, \pi'\}$, and vertex labels are $\Lambda = \Pi \cup \{\mathrm{ptype}(\pi), \mathrm{ptype}(\pi')\}$. The π-ports are $u_1, \ldots, u_4$ and the π'-ports are $v_1, \ldots, v_3$ in Figure 2 (a). We write $\emptyset$ for the empty partial function, that removes all port labels.

The domain of the *HR-algebra* is the set $\mathcal{G}^{\mathsf{HR}}(\Lambda, \Delta) \subseteq \mathcal{G}(\Lambda, \Lambda)$ of graphs G, written $\mathcal{G}^{\mathsf{HR}}$ when Λ and Δ are understood, where $\lambda_\mathsf{G}(v_1) \cap \lambda_\mathsf{G}(v_2) \cap \Pi \neq \emptyset$ implies $v_1 = v_2$, for any two vertices $v_1, v_2 \in \mathsf{V}_\mathsf{G}$. In other words, a port labels at most one vertex. In the usual terminology, ports are called *sources*, when the HR-algebra is understood from the context. The operations of HR are the following, see Figure 3 (b) for an illustration:

- *Composition*: $\mathsf{G}_1 \parallel_{\mathbf{p}_1,\mathbf{p}_2} \mathsf{G}_2 \stackrel{\text{def}}{=} H$ iff H is obtained from the disjoint union of $\mathsf{G}_1, \mathsf{G}_2 \in \mathcal{G}^{\mathsf{HR}}$ of sorts $\mathbf{p}_1$ and $\mathbf{p}_2$, respectively (taking an isomorphic copy of G_2 disjoint from G_1, if necessary) by joining all pairs of vertices $v_i \in \mathsf{V}_{\mathsf{G}_i}$, for $i = 1, 2$ such that $\lambda_{\mathsf{G}_1}(v_1) \cap \lambda_{\mathsf{G}_2}(v_2) \cap \Pi \neq \emptyset$. The label of a vertex v obtained by joining $v_1 \in \mathsf{V}_{\mathsf{G}_1}$ with $v_2 \in \mathsf{V}_{\mathsf{G}_2}$ is $\lambda_{\mathsf{G}_1}(v_1) \cup \lambda_{\mathsf{G}_2}(v_2)$. Since the label of a vertex in a graph from $\mathcal{G}^{\mathsf{HR}}$ contains at most one source, the composition of any two graphs from $\mathcal{G}^{\mathsf{HR}}$ is contained in $\mathcal{G}^{\mathsf{HR}}$. We write $\parallel$ instead of $\parallel_{\mathbf{p}_1,\mathbf{p}_2}$ when the argument sorts are understood.

- *Source redefinition*: relab_α is defined in the same way as for VR, with the further restriction that α is an injective partial function. This restriction is necessary to ensure that $\mathsf{relab}_\alpha(\mathsf{G}) \in \mathcal{G}^{\mathsf{HR}}$ if $\mathsf{G} \in \mathcal{G}^{\mathsf{HR}}$.
- *Single-vertex graphs*: are defined in the same way as for VR.
- *Single-edge graphs*: for an edge label $\delta \in \Delta$ and two port labels $\pi_1 \neq \pi_2 \in \Pi$, $\overrightarrow{\delta}_{\pi_1,\pi_2} \stackrel{\mathsf{def}}{=} H$ iff $\mathsf{V}_H = \{v_1, v_2\}$, $\mathsf{E}_H = \{(v_1, \delta, v_2)\}$, $\lambda_H(v_i) = \{\pi_i, \mathrm{ptype}(\pi_i)\}$, for $i = 1, 2$.

HR-terms are defined just as VR-terms and $\mathrm{val}^{\mathsf{HR}}(\theta)$ denotes the graph obtained by interpreting the function symbols in the ground term θ as the corresponding HR-operations.

Example 2. We write (π) for the partial function that acts as the identity on $\{\pi\}$, erasing all source labels except π. The HR-term $\theta' \stackrel{\mathsf{def}}{=} \mathsf{relab}_\emptyset(\overrightarrow{a}_{\pi_\varepsilon,\pi'_\varepsilon}) \parallel (\|_{i\in 1,\ldots,4} \mathsf{relab}_{(\pi_\varepsilon)}(\overrightarrow{\varepsilon}_{\pi,\pi_\varepsilon})) \parallel (\|_{j\in 1,\ldots,3} \mathsf{relab}_{(\pi'_\varepsilon)}(\overrightarrow{\varepsilon}_{\pi',\pi'_\varepsilon}))$ evaluates to the graph shown in Figure 2 (b). The edge labels are $\Delta = \{a, \varepsilon\}$, the port labels are $\Pi = \{\pi, \pi', \pi_\varepsilon, \pi'_\varepsilon\}$, the vertex labels are $\Lambda = \Pi \cup \{\mathrm{ptype}(\pi), \mathrm{ptype}(\pi'), \mathrm{ptype}(\pi_\varepsilon), \mathrm{ptype}(\pi'_\varepsilon)\}$. The π, π', π_ε and π'_ε-ports in Figure 2 are respectively $u_{1,\ldots,4}$, $v_{1,\ldots,3}$, u_ε, v_ε.

2.2 Grammars

Graph grammars are a standard way to represent sets of graphs. Let Ω be an algebra, either VR or HR. A Ω-*grammar* 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) $X \to t[X_1, \ldots, X_n]$, where $X, X_1, \ldots, X_n \in \Xi$ are nonterminals and t is a Ω-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 Ω-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 t, for some rule $X \to t[X_1, \ldots, X_n]$ of Γ. A X-*derivation* is a sequence of steps starting with a nonterminal X. The derivation is *complete* if it ends with a ground term. Let $\mathcal{L}_X(\Gamma) \stackrel{\mathsf{def}}{=} \{\mathrm{val}^\Omega(\theta) \mid X \Rightarrow^*_\Gamma \theta \text{ is a complete derivation}\}$ and $\mathcal{L}(\Gamma) \stackrel{\mathsf{def}}{=} \bigcup_{\to X \in \Pi} \mathcal{L}_X(\Gamma)$ be the *language* of Γ, the algebra Ω being understood from the rules of Γ. A set of graphs is VR (*resp.* HR) iff it is the language of a VR (*resp.* HR) grammar.

Example 3. The VR-grammar Γ below produces the term θ from Example 1. The language of this grammar contains Figure 2 (a). The HR-grammar Γ' produces the term θ' from Example 2. The language of this grammar contains Figure 2 (b).

$$\Gamma : \begin{cases} \to S \\ S \to \mathsf{relab}_\emptyset(\mathsf{add}_{a,\pi,\pi'}(K)) \\ K \to \bullet\pi \\ K \to \bullet\pi' \\ K \to K \oplus K \end{cases} \qquad \Gamma' : \begin{cases} \to S \\ S \to \mathsf{relab}_\emptyset(\overrightarrow{a}_{\pi_\varepsilon,\pi'_\varepsilon} \parallel K) \\ K \to \mathsf{relab}_{(\pi_\varepsilon)}(\overrightarrow{\varepsilon}_{\pi,\pi_\varepsilon}) \\ K \to \mathsf{relab}_{(\pi'_\varepsilon)}(\overrightarrow{\varepsilon}_{\pi',\pi'_\varepsilon}) \\ K \to K \parallel K \end{cases}$$

It is known that the expressivity of VR-grammars strictly subsumes that of HR-grammars [15]. The relation between VR and HR is made precise by the following:

Definition 1. *Let $\mathcal{E}$ be a set of edge labels disjoint from Δ. An $\mathcal{E}$-graph is a graph $\mathsf{G} \in \mathcal{G}(\Lambda, \Delta \cup \mathcal{E} \cup \overleftarrow{\mathcal{E}})$ such that*

- *$\overleftarrow{\mathcal{E}} \stackrel{\mathsf{def}}{=} \{\overleftarrow{e} \mid e \in \mathcal{E}\}$, and for every edge (u, e, v), $e \in \mathcal{E}$, there exists an edge $(v, \overleftarrow{e}, u)$.*

- *the subgraph of* G *consisting of* $\mathcal{E}$*-labeled edges and the vertices that have incoming* $\mathcal{E}$*-labeled edges is a forest, i.e., every non-root vertex points to a single parent.*

The expansion *of an* $\mathcal{E}$*-graph* $\mathsf{G} \in \mathcal{G}(\Lambda, \Delta \cup \mathcal{E} \cup \overleftarrow{\mathcal{E}})$ *is* $\exp(\mathsf{G}) \in \mathcal{G}(\Lambda, \Delta)$*, where:*
- $\mathsf{V}_{\exp(\mathsf{G})}$ *is the set of vertices of* G *that are not the target of an* $\mathcal{E}$*-labeled edge,*
- $\mathsf{E}_{\exp(\mathsf{G})}$ *is the set of edges* (v_1, δ, v_2) *for which there exist* $\mathcal{E}$*-labeled paths from* v_i *to some vertices* $v_i' \in \mathsf{G}$*, for* $i = 1, 2$*, such that* $(v_1', \delta, v_2') \in \mathsf{E}_\mathsf{G}$*, for some* $\delta \in \Delta$*,*
- $\lambda_{\exp(\mathsf{G})}(v) \stackrel{\text{def}}{=} \lambda_\mathsf{G}(v)$.

Proposition 1 (Proposition 2.4 in [12]). *For each VR-grammar* Γ*, one can build a HR-grammar* Γ' *such that* $\mathcal{L}(\Gamma) = \exp(\mathcal{L}(\Gamma'))$.

Example 4. The HR-grammar Γ' from Example 3 is obtained by transformation of the VR-grammar Γ from the same example, such that $\mathcal{L}(\Gamma) = \exp(\mathcal{L}(\Gamma'))$, where $\mathcal{E} = \{\varepsilon\}$. We omit $\overleftarrow{\mathcal{E}}$-edges from this figure: their placement is simply the inverse of $\mathcal{E}$-edges.

3 Parameterized Systems

We aim at describing a family of networks by a VR-grammar. To model the behavior of a network, we associate to each vertex in the graph a *process type*, which is a finite-state machine represented as a Petri net of a particular form. We denote by $\mathbb{P}$ the fixed and finite set of process types. In the rest of this paper, the alphabet of vertex labels is assumed to be $\Lambda \stackrel{\text{def}}{=} \mathbb{P} \uplus \Pi$, where $\Pi = \biguplus_{\mathsf{p} \in \mathbb{P}} \Pi_\mathsf{p}$ is a set of port labels partitioned according to the process types, *i.e.,* Π_p is the set of ports corresponding to the process type p. The intuition is that different vertices labeled with a process type run copies of that process type, called *processes*. Neighbouring processes communicate by joining their transitions in a symmetric synchronous rendezvous, represented by a pair of transitions, that labels the edge between their host vertices. Hence, the alphabet Δ of edge labels is considered to be the set of pairs of transitions from the process types $\mathbb{P}$.

3.1 Petri Nets

We make the definition of process types precise by recalling Petri nets. A *net* is a tuple $\mathsf{N} = (\mathsf{Q}_\mathsf{N}, \mathsf{T}_\mathsf{N}, \mathsf{W}_\mathsf{N})$, where Q_N is a finite set of *places*, T_N is a finite set of *transitions*, disjoint from Q_N, and $\mathsf{W}_\mathsf{N} : (\mathsf{Q}_\mathsf{N} \times \mathsf{T}_\mathsf{N}) \cup (\mathsf{T}_\mathsf{N} \times \mathsf{Q}_\mathsf{N}) \to \mathbb{N}$ is a *weighted incidence relation* between places and transitions. For all $x, y \in \mathsf{Q}_\mathsf{N} \cup \mathsf{T}_\mathsf{N}$ such that $\mathsf{W}_\mathsf{N}(x,y) > 0$, we say that there is an *edge of weight* $\mathsf{W}_\mathsf{N}(x,y)$ between x and y. For an element $x \in \mathsf{Q}_\mathsf{N} \cup \mathsf{T}_\mathsf{N}$, we define the set of *predecessors* ${}^\bullet x \stackrel{\text{def}}{=} \{y \in \mathsf{Q}_\mathsf{N} \cup \mathsf{T}_\mathsf{N} \mid \mathsf{W}_\mathsf{N}(y,x) > 0\}$, *successors* $x^\bullet \stackrel{\text{def}}{=} \{y \in \mathsf{Q}_\mathsf{N} \cup \mathsf{T}_\mathsf{N} \mid \mathsf{W}_\mathsf{N}(x,y) > 0\}$ and predecessor-successor pair ${}^\bullet x^\bullet \stackrel{\text{def}}{=} ({}^\bullet x, x^\bullet)$. If not obvious from the context, we will specify the net in which the predecessor and successor and considered: $({}^\bullet x)_\mathsf{N}$, $(x^\bullet)_\mathsf{N}$, $({}^\bullet x^\bullet)_\mathsf{N}$.

A *marking* of N is a function $\mathsf{m} : \mathsf{Q}_\mathsf{N} \to \mathbb{N}$. A transition t is *enabled* in the marking m if $\mathsf{m}(q) \geq \mathsf{W}_\mathsf{N}(q,t)$, for each place $q \in \mathsf{Q}$. For all markings m, m' and transitions $t \in \mathsf{T}$, we write $\mathsf{m} \stackrel{t}{\rightsquigarrow} \mathsf{m}'$ whenever t is enabled in m and $\mathsf{m}'(q) = \mathsf{m}(q) - \mathsf{W}_\mathsf{N}(q,t) + \mathsf{W}_\mathsf{N}(t,q)$, for all $q \in \mathsf{Q}_\mathsf{N}$. Given a marking m, a sequence of transitions $\rho = (t_1, t_2, \ldots) \in (\mathsf{T}_\mathcal{N})^\infty$ is

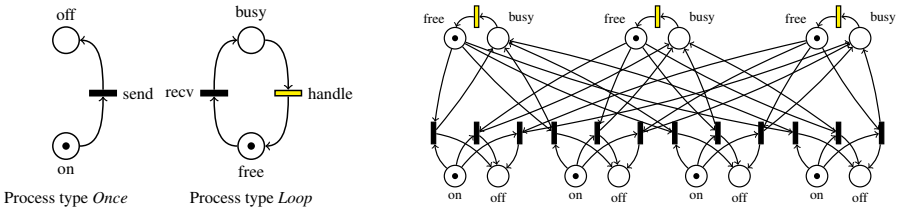


Fig. 4. Examples of process types (a). The behavior of a $\overrightarrow{K}_{4,3}(\text{send},\text{recv})$ system (b)

a *firing sequence* iff either (i) ρ is empty, or (ii) there exist markings $\mathsf{m}_1, \mathsf{m}_2, \ldots$ such that $\mathsf{m} \overset{t_1}{\rightsquigarrow} \mathsf{m}_1 \overset{t_2}{\rightsquigarrow} \mathsf{m}_2 \overset{t_3}{\rightsquigarrow} \cdots$. When ρ has finite length n, we can write $\mathsf{m} \overset{\rho}{\rightsquigarrow} \mathsf{m}_n$, and we say that ρ is a firing sequence from m to m_n.

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 $\mathsf{Q}_\mathcal{N} \overset{\text{def}}{=} \mathsf{Q}_\mathsf{N}$, $\mathsf{T}_\mathcal{N} \overset{\text{def}}{=} \mathsf{T}_\mathsf{N}$, $\mathsf{W}_\mathcal{N} \overset{\text{def}}{=} \mathsf{W}_\mathsf{N}$ and $\text{init}_\mathcal{N} \overset{\text{def}}{=} \mathsf{m}_0$ for the elements of $\mathcal{N}$. A marking m is *reachable* in $\mathcal{N}$ iff there exists a finite firing sequence ρ such that $\mathsf{m}_0 \overset{\rho}{\rightsquigarrow} \mathsf{m}$. We denote by $\mathit{Paths}(\mathcal{N})$ the set of finite or infinite firing sequences of $\mathcal{N}$ starting from $\text{init}_\mathcal{N}$.

3.2 Behaviors

We formalize the behavior of a system (*i.e.,* a graph whose vertices are labeled by process types) using PNs. To do so, we associate to each process type a PN having a special form:

Definition 2. *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.,* $\mathsf{T}_\mathsf{p} = \mathsf{T}_\mathsf{p}^{obs} \uplus \mathsf{T}_\mathsf{p}^{int}$*, such that each transition has exactly one predecessor and one successor. Let* $\mathbb{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_k\}$ *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, every reachable marking of a process type has exactly one token. We denote by $\mathsf{Q}_\mathbb{P} \overset{\text{def}}{=} \biguplus_{\mathsf{p}\in\mathbb{P}} \mathsf{Q}_\mathsf{p}$ and $\mathsf{T}_\mathbb{P}^{obs} \overset{\text{def}}{=} \biguplus_{\mathsf{p}\in\mathbb{P}} \mathsf{T}_\mathsf{p}^{obs}$ the sets of places and observable transitions from some $\mathsf{p} \in \mathbb{P}$, respectively.

Example 5. Figure 4 (a) shows two examples of process types, *Once* and *Loop*. Observable transitions are shown in black, and internal transitions in yellow. They have $\mathsf{Q}_{Once} = \{\text{on}, \text{off}\}$, $\mathsf{T}_{Once}^{obs} = \mathsf{T}_{Once} = \{\text{send}\}$, $\mathsf{Q}_{Loop} = \{\text{free}, \text{busy}\}$, $\mathsf{T}_{Loop}^{obs} = \{\text{recv}\}$ and $\mathsf{T}_{Loop}^{int} = \{\text{handle}\}$.

Definition 3. *A* system $\mathsf{S} = (\mathsf{V}_\mathsf{S}, \mathsf{E}_\mathsf{S}, \lambda_\mathsf{S}) \in \mathcal{G}(\mathbb{P} \uplus \Pi, \mathsf{T}_\mathbb{P}^{obs} \times \mathsf{T}_\mathbb{P}^{obs})$ *is a graph whose vertices* $v \in \mathsf{V}_\mathsf{S}$ *are labeled with exactly one process type* $\text{proc}_\mathsf{S}(v) = \mathsf{p} \overset{\text{def}}{\iff} \lambda_\mathsf{S}(v) \cap \mathbb{P} = \{\mathsf{p}\}$ *and at most one port* $\text{port}_\mathsf{S}(v) = \pi \overset{\text{def}}{\iff} \lambda_\mathsf{S}(v) \cap \Pi = \{\pi\}$*. Edges* $v_1 \xrightarrow{(t_1,t_2)} v_2 \in \mathsf{E}_\mathsf{S}$ *are labeled with pairs of observable transitions, such that* $t_i \in \mathsf{T}_{\text{proc}_\mathsf{S}(v_i)}^{obs}$*, for* $i = 1,2$*. We denote by* $\mathcal{S}(\mathbb{P}) \overset{\text{def}}{=} \mathcal{G}(\mathbb{P}, \mathsf{T}_\mathbb{P}^{obs} \times \mathsf{T}_\mathbb{P}^{obs})$ *the set of systems of empty sort, i.e., without ports.*

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

Definition 4. *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 S *is* $\beta(\mathsf{S}) \stackrel{\text{def}}{=} (\mathsf{N}, \mathsf{m}_0)$*, where:*

- $\mathsf{Q}_{\mathsf{N}} \stackrel{\text{def}}{=} \{(q,v) \mid q \in \mathsf{Q}_{\text{proc}_{\mathsf{S}}(v)},\ v \in \mathsf{V}_{\mathsf{S}}\}$*, a place (q,v) corresponds to the place q of the process type that labels the vertex v,*
- $\mathsf{T}_{\mathsf{N}} \stackrel{\text{def}}{=} \mathsf{E} \cup \{(t,v) \mid t \in \mathsf{T}^{int}_{\text{proc}_{\mathsf{S}}(v)},\ v \in \mathsf{V}_{\mathsf{S}}\}$*, 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 that labels the vertex v,*
- *the weight function W_{N} is such that internal transitions are preserved, and observable transitions are merged according to edge labels. That is, for every v such that $({}^\bullet t^\bullet)_{\text{proc}_{\mathsf{S}}(v)} = (\{q\}, \{q'\})$ we have $({}^\bullet (t,v)^\bullet)_{\mathsf{N}} = (\{(q,v)\}, \{(q',v')\})$, and for every edge $e = (v \xrightarrow{(t,t')} v')$ with $({}^\bullet t^\bullet)_{\text{proc}_{\mathsf{S}}(v)} = (\{q_1\}, \{q_2\})$ and $({}^\bullet t'^\bullet)_{\text{proc}_{\mathsf{S}}(v')} = (\{q'_1\}, \{q'_2\})$ we have $({}^\bullet e^\bullet)_{\mathsf{N}} = (\{(q_1,v),(q'_1,v')\}, \{(q_2,v),(q'_2,v')\})$.*
- $\mathsf{m}_0(q,v) \stackrel{\text{def}}{=} \text{init}_{\text{proc}_{\mathsf{S}}(v)}(q)$*, for all $v \in \mathsf{V}_{\mathsf{S}}$ and $q \in \mathsf{Q}_{\text{proc}_{\mathsf{S}}(v)}$.*

Example 6. Figure 4 (b) shows the behavior of $\overrightarrow{K}_{4,3}(\text{send}, \text{recv})$, having the architecture shown in Figure 2 (a), with the red vertices $u_{1\ldots4}$ labeled with the process type *Once* and blue vertices $v_{1\ldots3}$ labeled with the process type *Loop*, from Figure 4. The edges are labeled by the pair $(\text{send}, \text{recv})$ of observable transitions from *Once* and *Loop*.

3.3 The Parametric Reachability Problem

A *parametric* VR (*resp.* HR) *system* is described by a VR (*resp.* HR) grammar Γ. We define our decision problem as a parametric reachability problem asking if there exists an instance of a parametric system, described by a grammar, whose behavior reaches an error configuration from a given set. If the answer is negative, we have a proof that the parametric system is correct.

Let $\mathbb{X}$ be a fixed countably infinite set of variables, interpreted over natural numbers. An arithmetic formula α is a (possibly quantified) first-order formula using variables in $\mathbb{X}$, constants denoting natural numbers, the binary operations of addition and multiplication and the (in)equality relation. We denote by $\mathbb{A}$ the set of such formulæ. Let $\mathcal{N}$ be a PN and $\ell : \mathsf{Q}_{\mathcal{N}} \rightharpoonup \mathbb{X}$ be a partial function that labels certain places of $\mathcal{N}$ with variables. The boolean value $[\![\alpha]\!]^{\ell}_{\mathsf{m}} \in \{\text{true}, \text{false}\}$ of an arithmetic formula α in a marking $\mathsf{m} : \mathsf{Q}_{\mathcal{N}} \to \mathbb{N}$ is obtained by replacing each variable x that occurs free in α by the integer value $\sum_{q \in \ell^{-1}(x)} \mathsf{m}(q)$, *i.e.,* the total number of tokens from the places associated with x in m. We further define $\mathit{Atoms}^{\ell}(\mathsf{m}) \stackrel{\text{def}}{=} \{\alpha \in \mathbb{A} \mid [\![\alpha]\!]^{\ell}_{\mathsf{m}} = \text{true}\}$, *i.e.,* the set of arithmetic formulæ satisfied by a given marking and a variable labeling. A PN $\mathcal{N}$ satisfies a reachability specification α for a place labeling ℓ, written $(\mathcal{N}, \ell) \models \alpha$, iff there exists a finite firing sequence ρ leading to a reachable marking $\text{init}_{\mathcal{N}} \stackrel{\rho}{\rightsquigarrow} \mathsf{m}'$ such that $\alpha \in \mathit{Atoms}^{\ell}(\mathsf{m}')$.

We specialize the satisfiability of a reachability specification by an arbitrary PN to the satisfiability by the behavior of a system $\mathsf{S} \in \mathcal{S}(\mathbb{P})$ and a labeling $\mathsf{L} : \mathsf{Q}_{\mathbb{P}} \rightharpoonup \mathbb{X}$ of the

places from the process types $\mathbb{P}$ with variables:

$$(\mathsf{S},\mathsf{L}) \models \alpha \stackrel{\mathsf{def}}{\iff} (\beta(\mathsf{S}),\mathsf{L}\circ\downarrow_1) \models \alpha$$

where $\downarrow_1$ is the projection of a pair on its first component, *i.e.,* $\mathsf{L}\circ\downarrow_1$ labels each pair $(q,v)\in\mathsf{Q}_{\beta(\mathsf{S})}$ by the variable $\mathsf{L}(q)$, if the latter is defined. Intuitively, we require of ℓ that it assigns the same variable to all "copies" of the same place $\{(q,v) \mid v\in\mathsf{V}_\mathsf{S}\}$. Hence, the boolean value $[\![\alpha]\!]_\mathsf{m}^{\mathsf{L}\circ\downarrow_1}$ of an atomic proposition in a marking m of $\beta(\mathsf{S})$ is obtained by mapping each free variable x from α to the total number of tokens from a place $(q,v)\in\mathsf{Q}_{\beta(\mathsf{S})}$ such that $\mathsf{L}(q)=x$. This paper is concerned with the following problem:

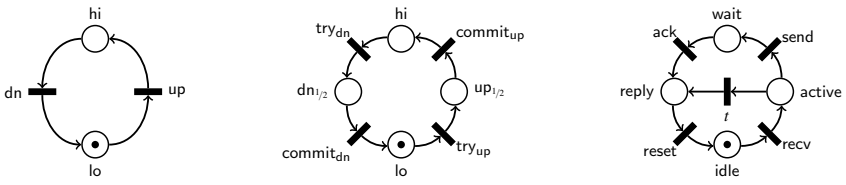
Definition 5. *Let $\mathbb{P}$ be a set of process types. The Parametric Reachability Problem $\mathsf{PRP}_\mathbb{P}(\Gamma,\alpha,\mathsf{L})$ takes as input a grammar Γ such that $\mathcal{L}(\Gamma)\subseteq\mathcal{S}(\mathbb{P})$, a reachability formula α and a place labeling $\mathsf{L}:\mathsf{Q}_\mathbb{P}\rightharpoonup\mathbb{X}$, and asks if there exists some system $\mathsf{S}\in\mathcal{L}(\Gamma)$ for which $(\mathsf{S},\mathsf{L})\models\alpha$ holds.*

Example 7. Choosing Γ the VR-grammar (Example 3), the set of variables $\mathbb{X}\stackrel{\mathsf{def}}{=}\{x,y\}$, the place labeling $\mathsf{L}=[\mathsf{on}\mapsto x,\mathsf{off}\mapsto y]$ defined on the process types from Figure 4 (a), and the formula $\alpha\stackrel{\mathsf{def}}{=}y\geq x+2$, we find that $\mathsf{PRP}_\mathbb{P}(\Gamma,\alpha,\mathsf{L})$ has a positive answer, because the behavior Figure 4 (b) which belongs to $\beta(\mathcal{L}(\Gamma))$ admits a reachable marking m with 3 tokens on off and 1 token on on, satisfying $[\![\alpha]\!]_\mathsf{m}^{\mathsf{L}\circ\downarrow_1}=[\![3\geq 1+2]\!]=\text{true}$.

In the light of inherent theoretical boundaries, related to the undecidability of the above parametric verification problem [14], several semi-algorithmic methods have been proposed, for parametric HR systems [9] and parametric systems described using similar formalisms, such as a dialect of separation logic with inductive definitions [10] and a recursive term algebra [8]. In particular, the architecture description languages used in [10,8] have similar graph composition and relabeling as the more standard HR grammars. For these methods, the verification of certain coverability properties (*e.g.,* absence of deadlocks and critical section violations) relies on the generation of structural invariants for the parametric family of behaviors, directly from the specification of the architectures and the process types. Examples include *trap invariants*, *i.e.,* sets of places that, once marked, remain forever marked, and *mutex invariants*, *i.e.,* sets of places of which at most one is marked. The generation of structural invariants can be redefined for HR grammars at little cost.

In contrast, there is virtually no verification method for parametric VR systems. Many decidability results in the literature consider parametric systems with clique architectures [16], that can be described by simple VR-grammars. A prominent result is the decidability of the more general parametric model checking problem (PMCP) for *token-passing systems* (where communication is restricted to the passing of a token between processes) with MSO-definable bounded clique-width architectures [4, Theorem 26]. It is known that each MSO-definable set of bounded clique-width is definable by a VR grammar, but not viceversa. Moreover, an orthogonal[5] decidability result for parametric HR token-passing systems is given in [9, Theorem 4]. It is an interesting open problem whether this decidability result for HR systems carries over to VR systems, but this exceeds the scope of the current paper, and will be considered in future work.

[5] This result applies to all HR sets of architectures, not just the MSO-definable ones.



(a) (b) (c)

Fig. 5. A process type p (a), the corresponding process type $\mathsf{p}_{1/2}$ (b), and ε_t (c).

4 Translating VR to HR Systems

The existence of several verification techniques for parametric systems with HR architectures and the scarsity of similar results for VR systems motivates us to define a translation from VR to HR systems, that preserves the answer of the PRP decision problem (Definition 5). This translation uses expansion to encode dense graphs as sparse graphs (see Figure 2). Since labeled graphs are used as system models (Definition 3), the behavior of a sparse graph must be equivalent to the behavior of its expansion, on what concerns the satisfiability of reachability formulæ, thus reducing each instance of the PRP problem for a VR grammar to an instance of the same problem for an HR grammar.

Theorem 1. *There exist computable* $\mathbb{P}^{\mathcal{H}}$ *and* $\mathsf{L}^{\mathcal{H}}$*, process types and place labeling respectively, such that for any VR-grammar* Γ*, variable labeling* $\mathsf{L} : \mathsf{Q}_{\mathbb{P}} \to \mathbb{X}$ *and reachability formula* α*, one can effectively build a HR-grammar* Γ' *such that* $\mathsf{PRP}_{\mathbb{P}}(\Gamma, \alpha, \mathsf{L})$ $\iff \mathsf{PRP}_{\mathbb{P}^{\mathcal{H}}}(\Gamma', \alpha, \mathsf{L}^{\mathcal{H}})$

This result shows that it is possible to solve each instance of the parametric model checking problem that takes a VR grammar as input by an effective reduction to an instance of the same problem for a HR grammar, which, as previously mentioned, has received more attention lately [9].

Intuitively the translation works as follows: vertices in the HR system are either "real" vertices from the original VR system, or routing nodes. A unique vertex of the HR system will act as representative of all vertices currently π-ports in the VR system, and communication between a π-port v and a π'-port v' in the VR system will in the HR system roughly take the form of (1) a request from v to its representative v_π through a chain of routing nodes, (2) similarly a request from v' to its representative $v_{\pi'}$, (3) a direct communication between v_π and $v_{\pi'}$ (4) an acknowledgement from v_π to v following the inverse chain of routing nodes, (5) an acknowledgement from $v_{\pi'}$ to v'. This is a simplification because in reality steps (1–2) and (4–5) above may be interleaved, and rather than a single representative there is one representative per observable transition. We define below the process types that correspond to routing nodes, give an intuition of how they function, then show the actual inductive translation.

Let $\mathbb{P}$ be the set of process types used by the VR grammars, in the rest of this section. We define a new set of process types $\mathbb{P}^{\mathcal{H}}$ such that $\mathsf{Q}_{\mathbb{P}} \subseteq \mathsf{Q}_{\mathbb{P}^{\mathcal{H}}}$, see Figure 5 (b) for an example. First, let $\mathbb{P}_{1/2} \stackrel{\text{def}}{=} \{\mathsf{p}_{1/2} \mid \mathsf{p} \in \mathbb{P}\}$ be a set of process types disjoint from $\mathbb{P}$. A generic $\mathsf{p}_{1/2}$ is defined below from $\mathsf{p} \in \mathbb{P}$. Intuitively, $\mathsf{p}_{1/2}$ simulates the behavior of p in the context of communication through routing nodes: each observable transition is split into two halves, one sending a request, then eventually receiving the acknowledgement.

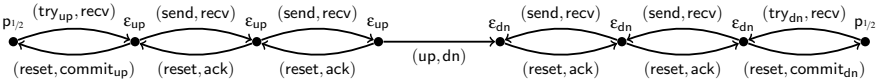


Fig. 6. An $\varepsilon_{\mathsf{up}}$-path meeting an $\varepsilon_{\mathsf{dn}}$-path to simulate the interaction $(\mathsf{up}, \mathsf{dn})$ between two vertices of type p from Figure 5 (a).

- $\mathsf{Q}_{\mathsf{p}_{1/2}} \stackrel{\text{def}}{=} \mathsf{Q}_{\mathsf{p}} \uplus \{t_{1/2} \mid t \in \mathsf{T}_{\mathsf{p}}^{obs}\}$, each $t_{1/2}$ is a new place associated with the transition t,
- $\mathsf{T}_{\mathsf{p}_{1/2}} \stackrel{\text{def}}{=} \mathsf{T}_{\mathsf{p}}^{int} \uplus \{\mathsf{try}_t, \mathsf{commit}_t \mid t \in \mathsf{T}_{\mathsf{p}}^{obs}\}$, *i.e.,* each observable transition t of p is split into try_t (an attempt at firing t) and commit_t (t has been fired remotely),
- $\mathsf{W}_{\mathsf{p}_{1/2}} \stackrel{\text{def}}{=} \mathsf{W}_{\mathsf{p}}\restriction_{\mathsf{Q}_{\mathsf{p}} \times \mathsf{T}_{\mathsf{p}}^{int} \uplus \mathsf{T}_{\mathsf{p}}^{int} \times \mathsf{Q}_{\mathsf{p}}} \uplus \{(q, \mathsf{try}_t, 1), (\mathsf{try}_t, t_{1/2}, 1) \mid q \in \mathsf{Q}_{\mathsf{p}},\ t \in \mathsf{T}_{\mathsf{p}}^{obs},\ \mathsf{W}_{\mathsf{p}}(q,t) = 1\} \uplus \{(t_{1/2}, \mathsf{commit}_t, 1), (\mathsf{commit}_t, q, 1) \mid t \in \mathsf{T}_{\mathsf{p}}^{obs},\ \mathsf{W}_{\mathsf{p}}(t,q) = 1\}$, *i.e.,* whenever t is an observable transition from q to q', there are observable transitions try_t from q to $t_{1/2}$ and commit_t from $t_{1/2}$ to q' in $\mathsf{p}_{1/2}$.
- $\mathrm{init}_{\mathsf{p}_{1/2}} \stackrel{\text{def}}{=} \mathrm{init}_{\mathsf{p}}$.

Second, for each observable transition $t \in \mathsf{T}_{\mathbb{P}}^{obs}$, we consider a new process type ε_t, defined below. See Figure 5 (c) for an example. This is what we call the "routing nodes".

- $\mathsf{Q}_{\varepsilon_t} \stackrel{\text{def}}{=} \{\mathsf{idle}_t, \mathsf{active}_t, \mathsf{wait}_t, \mathsf{reply}_t\}$,
- $\mathsf{T}_{\varepsilon_t} = \mathsf{T}_{\varepsilon_t}^{obs} \stackrel{\text{def}}{=} \{\mathsf{recv}, \mathsf{send}, \mathsf{ack}, \mathsf{reset}, t\}$,
- $\mathsf{W}_{\varepsilon_t}$ consists of the following edges, all of weight 1:

$$\bullet\mathsf{recv}\bullet \stackrel{\text{def}}{=} (\mathsf{idle}_t, \mathsf{active}_t) \qquad \bullet\mathsf{send}\bullet \stackrel{\text{def}}{=} (\mathsf{active}_t, \mathsf{wait}_t) \qquad \bullet t \bullet \stackrel{\text{def}}{=} (\mathsf{active}_t, \mathsf{reply}_t)$$
$$\bullet\mathsf{ack}\bullet \stackrel{\text{def}}{=} (\mathsf{wait}_t, \mathsf{reply}_t) \qquad \bullet\mathsf{reset}\bullet \stackrel{\text{def}}{=} (\mathsf{reply}_t, \mathsf{idle}_t)$$

- $\mathrm{init}_{\varepsilon_t}(\mathsf{idle}_t) \stackrel{\text{def}}{=} 1$ and $\mathrm{init}_{\varepsilon_t}(\mathsf{active}_t) = \mathrm{init}_{\varepsilon_t}(\mathsf{wait}_t) = \mathrm{init}_{\varepsilon_t}(\mathsf{reply}_t) \stackrel{\text{def}}{=} 0$.

Finally, $\mathbb{P}^{\mathcal{H}} \stackrel{\text{def}}{=} \mathbb{P}_{1/2} \cup \{\varepsilon_t \mid t \in \mathsf{T}_{\mathsf{p}}^{obs}\}$, "real vertices" and "routing nodes" respectively.

Example 8. Figure 5 (a) shows examples of p, $\mathsf{p}_{1/2}$, and ε_t. Instances of ε_t exist for $t \in \mathsf{T}_{\mathsf{p}}^{obs} = \{\mathsf{up}, \mathsf{dn}\}$. Two processes of type p communicate via a $(\mathsf{up}, \mathsf{dn})$ transition, which is encoded by a sequence of transitions, see Figure 6:

$$(\underbrace{\mathsf{try}_{\mathsf{up}}}_{\mathsf{p}_{1/2}}, \underbrace{\mathsf{recv}), (\mathsf{send}, \mathsf{recv}), \ldots, (\mathsf{send}, \mathsf{recv}), (\mathsf{up}}_{\varepsilon_{\mathsf{up}}}, \underbrace{\mathsf{dn}}_{\varepsilon_{\mathsf{dn}}}), \underbrace{(\mathsf{reset}, \mathsf{ack}), \ldots, (\mathsf{reset}, \mathsf{ack}), (\mathsf{reset}}_{\varepsilon_{\mathsf{up}}}, \underbrace{\mathsf{commit}_{\mathsf{up}}}_{\mathsf{p}_{1/2}})$$

This matches the earlier intuition: the leftmost process tries to execute up and sends a request through routers $\varepsilon_{\mathsf{up}}$. At the same time, the right process tries to execute dn and sends a request through routers $\varepsilon_{\mathsf{dn}}$. When both requests reach adjacent routers, the transition $(\mathsf{up}, \mathsf{dn})$ is jointly executed. Following that, an acknowledgement is sent back to the processes that initiated the communication as a sequence of $(\mathsf{reset}, \mathsf{ack})$.

The variable labeling $\mathsf{L} : \mathsf{Q}_{\mathbb{P}} \rightharpoonup \mathbb{X}$ induces the following labeling $\mathsf{L}^{\mathcal{H}} : \mathsf{Q}_{\mathbb{P}^{\mathcal{H}}} \rightharpoonup \mathbb{X}$:

$$\mathsf{L}^{\mathcal{H}}(q) \stackrel{\text{def}}{=} \mathsf{L}(q), \text{ for each } q \in \mathsf{Q}_{\mathbb{P}}$$
$$\mathsf{L}^{\mathcal{H}}(\mathsf{active}_t) \stackrel{\text{def}}{=} \mathsf{L}(\bullet t) \qquad \mathsf{L}^{\mathcal{H}}(\mathsf{reply}_t) \stackrel{\text{def}}{=} \mathsf{L}(t\bullet), \text{ for each } t \in \mathsf{T}_{\mathbb{P}}^{obs}$$
$$\text{undefined everywhere else}$$

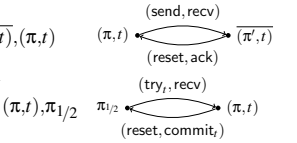
The rest of the paper is the proof of Theorem 1, instanciated with these $\mathbb{P}^{\mathcal{H}}$ and $\mathsf{L}^{\mathcal{H}}$.

4.1 Proof of Theorem 1

The main idea is to use the ε_t processes to define, for a VR-term θ, a HR-term $\mathcal{H}(\theta)$ that evaluates to a sparse system, from which the system $\mathrm{val}^{\mathsf{VR}}(\theta)$ can be retrived using graph expansion, *i.e.,* $\mathrm{val}^{\mathsf{VR}}(\theta) = \exp(\mathrm{val}^{\mathsf{HR}}(\mathcal{H}(\theta)))$ (Definition 1). Using the set of port labels $\Pi^{\mathcal{H}} \stackrel{\mathsf{def}}{=} \{\pi_{1/2}, (\pi,t), \overline{(\pi,t)} \mid \pi \in \Pi, t \in \mathsf{T}^{obs}_{\mathrm{ptype}(\pi)}\}$ where the process type of $\pi_{1/2}$ is $(\mathrm{ptype}(\pi))_{1/2}$ and the process type of (π,t) and $\overline{(\pi,t)}$ is ε_t, we define $\mathcal{H}(\theta)$ inductively on the structure of θ. That is, in addition to a source label $\pi_{1/2}$ for each port label $\pi \in \Pi$ that occurs in θ, we add two copies of each port label for each observable transition. Intuitively, (π,t) will be the port label denoting the root of the tree of ε_t processes, and $\overline{(\pi,t)}$ will be used as backup when we need a fresh copy of (π,t). We define a renaming function $\mathsf{fresh} : \Pi^{\mathcal{H}} \rightharpoonup \Pi^{\mathcal{H}}$ by $\mathsf{fresh}(\overline{(\pi,t)}) = (\pi,t)$, and undefined everywhere else. That way fresh substitutes each source for its copy and forgets the original one. We also use the renaming function $\mathsf{obs} : \Pi^{\mathcal{H}} \rightharpoonup \Pi^{\mathcal{H}}$ which is the identity on $\Pi \times \mathsf{T}^{obs}_{\mathbb{P}}$ and undefined everywhere else, *i.e.,* obs forgets every port label except the (π,t) labels.

Concretely, creating new edges between ε_t-vertices is done by the below functions:

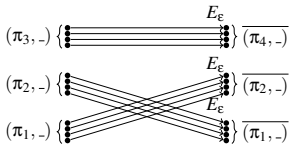
$$E^t_\varepsilon(\pi,\pi') \stackrel{\mathsf{def}}{=} \overrightarrow{(\mathsf{send},\mathsf{recv})}_{(\pi,t),\overline{(\pi',t)}} \parallel \overrightarrow{(\mathsf{reset},\mathsf{ack})}_{\overline{(\pi',t)},(\pi,t)} \qquad E^t_{1/2}(\pi) \stackrel{\mathsf{def}}{=} \overrightarrow{(\mathsf{try}_t,\mathsf{recv})}_{\pi_{1/2},(\pi,t)} \parallel \overrightarrow{(\mathsf{reset},\mathsf{commit}_t)}_{(\pi,t),\pi_{1/2}}$$



Recall that the main invariant during the translation is that whenever v in the VR system is a π-port, its corresponding vertex in the HR system is linked to the unique representative of (π,t) through a chain of routers ε_t.

An essential gadget used by the construction is the encoding of a relabeling function $\alpha : \Pi \rightharpoonup \Pi$ (which is by definition type-preserving) by a graph denoted $\mathsf{enc}(\alpha)$. We show an example of $\mathsf{enc}(\alpha)$, where α is the function $[\pi_1 \mapsto \pi_2, \pi_2 \mapsto \pi_1, \pi_3 \mapsto \pi_4]$. So as to not clutter the figure, we simply write E_ε to label the entire bundle of edges $\{E^t_\varepsilon \mid t \in \mathsf{T}^{obs}_{\mathbb{P}}\}$ and we write $(\pi,\_)$ for $\{(\pi,t) \mid t \in \mathsf{T}^{obs}_{\mathrm{ptype}(\pi)}\}$.

$$\mathsf{enc}(\alpha) \stackrel{\mathsf{def}}{=} \Big\|_{\substack{\pi,\pi' \in \Pi \\ \alpha(\pi)=\pi'}} \Big\|_{t \in \mathsf{T}^{obs}_{\mathrm{ptype}(\pi)}} E^t_\varepsilon(\pi,\pi')$$



Intuitively, as the renaming α turns every π-port into an $\alpha(\pi)$-port, $\mathsf{enc}(\alpha)$ links the previous representatives of $(\pi,\_)$ to the new representatives of $(\alpha(\pi),\_)$, preserving the invariant. The translation $\mathcal{H}$ is defined inductively on the structure of the term θ:

$$\begin{aligned}
\mathcal{H}(\bullet\pi) &\stackrel{\mathsf{def}}{=} \mathsf{relab}_{\mathsf{obs}}\Big(\bullet\pi_{1/2} \parallel \Big(\|_{t \in \mathsf{T}^{obs}_{\mathrm{ptype}(\pi)}} E^t_{1/2}(\pi)\Big)\Big) && \text{Figure 7 (a)}\\
\mathcal{H}(\mathsf{add}_{(t,t'),\pi,\pi'}(\theta_1)) &\stackrel{\mathsf{def}}{=} \overrightarrow{(t,t')}_{(\pi,t),(\pi',t')} \parallel \mathcal{H}(\theta_1) && \text{Figure 7 (b)}\\
\mathcal{H}(\mathsf{relab}_\alpha(\theta_1)) &\stackrel{\mathsf{def}}{=} \mathsf{relab}_{\mathsf{fresh}}\Big(\mathcal{H}(\theta_1) \parallel \mathsf{enc}(\alpha)\Big) && \text{Figure 7 (c,c',c")}\\
\mathcal{H}(\theta_1 \oplus_{\mathbf{p}_1,\mathbf{p}_2} \theta_2) &\stackrel{\mathsf{def}}{=} \mathsf{relab}_{\mathsf{fresh}}\left(\mathcal{H}(\theta_1) \parallel \mathsf{enc}(\mathrm{id}_{\mathbf{p}_1 \cap \mathbf{p}_2})\right) && \\
&\quad \parallel \mathsf{relab}_{\mathsf{fresh}}\left(\mathcal{H}(\theta_2) \parallel \mathsf{enc}(\mathrm{id}_{\mathbf{p}_1 \cap \mathbf{p}_2})\right) && \text{Figure 7 (d,d',d")}
\end{aligned}$$

These are quite straightforward once we understand the invariant: the translation of $\bullet\pi$

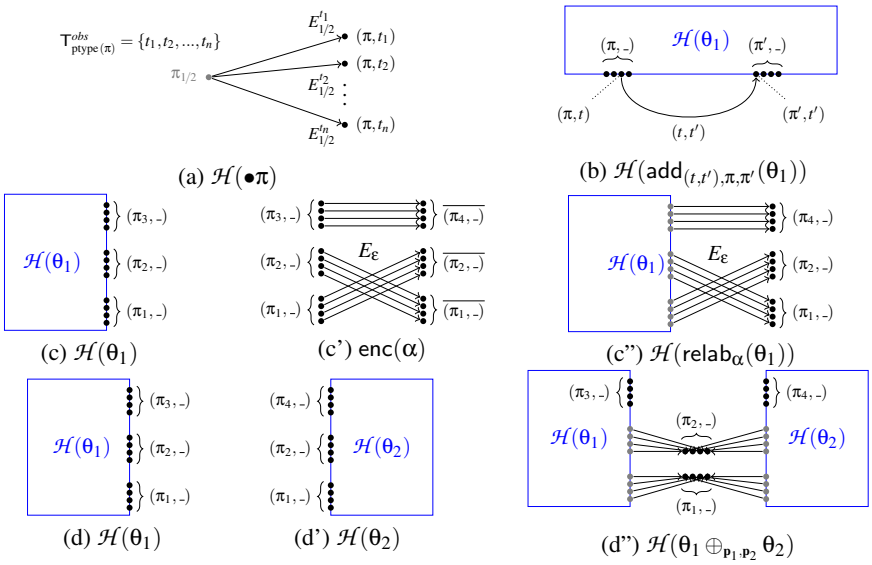


Fig. 7. Construction of $\mathcal{H}$ for each VR symbol. Relabeling uses $\alpha = [\pi_1 \mapsto \pi_2, \pi_2 \mapsto \pi_1, \pi_3 \mapsto \pi_4]$, and composition uses $\mathbf{p}_1 = \{\pi_1, \pi_2, \pi_3\}$ and $\mathbf{p}_2 = \{\pi_1, \pi_2, \pi_4\}$. Ports in gray are hidden during a relabeling.

creates a new vertex and links it to all relevant routing nodes, the rule for $\mathsf{add}_{(t,t'),\pi,\pi'}$ establishes a communication between the representatives of (π,t) and (π',t'), the one for $\oplus$ creates a unique common representative, and the relabeling is as explained earlier.

The relation between the systems $\mathsf{val}^{\mathsf{VR}}(\theta)$ and $\mathsf{val}^{\mathsf{HR}}(\mathcal{H}(\theta))$ is captured by the below lemma. Its proof is the same as in [12, Proposition 2.4], where the new edges have the same label, instead of the following sets of edge labels (Definition 1):

$$\mathcal{E} \stackrel{\mathsf{def}}{=} \{(\mathsf{try}_t, \mathsf{recv}) \mid t \in \mathsf{T}^{obs}_{\mathbb{P}}\} \cup \{(\mathsf{send}, \mathsf{recv})\}$$
$$\overleftarrow{\mathcal{E}} \stackrel{\mathsf{def}}{=} \{(\mathsf{reset}, \mathsf{commit}_t) \mid t \in \mathsf{T}^{obs}_{\mathbb{P}}\} \cup \{(\mathsf{reset}, \mathsf{ack})\}$$

We denote $(\mathsf{commit}_t, \mathsf{reset}) \stackrel{\mathsf{def}}{=} \overleftarrow{(\mathsf{try}_t, \mathsf{recv})}$ and $(\mathsf{ack}, \mathsf{reset}) \stackrel{\mathsf{def}}{=} \overleftarrow{(\mathsf{send}, \mathsf{recv})}$ (Definition 1). The meaning and orientation of these edges are shown on an example in Figure 6.

Lemma 1. *Let θ be a ground VR-term, $\mathsf{S} \stackrel{\mathsf{def}}{=} \mathsf{val}^{\mathsf{VR}}(\theta)$ and $\mathsf{S}' \stackrel{\mathsf{def}}{=} \mathsf{val}^{\mathsf{HR}}(\mathcal{H}(\theta))$. Then, $\mathsf{S} = \exp(\mathsf{S}')$. For each $v, v' \in \mathsf{V}_{\mathsf{S}'}$, if $E \neq \emptyset$ is the set of edges from v to v', then exactly one of the following holds:*

1. $(\mathsf{proc}_{\mathsf{S}'}(v), \mathsf{proc}_{\mathsf{S}'}(v')) = (\mathsf{p}_{1/2}, \varepsilon_t)$, $E = \{(\mathsf{try}_t, \mathsf{recv}), (\mathsf{commit}_t, \mathsf{reset})\}$ *and* $t \in \mathsf{T}^{obs}_{\mathsf{p}}$;
2. $(\mathsf{proc}_{\mathsf{S}'}(v), \mathsf{proc}_{\mathsf{S}'}(v')) = (\varepsilon_t, \varepsilon_{t'})$ *and* $E = \{(t,t')\}$;
3. $\mathsf{proc}_{\mathsf{S}'}(v) = \mathsf{proc}_{\mathsf{S}'}(v') = \varepsilon_t$ *and* $E = \{(\mathsf{send}, \mathsf{recv}), (\mathsf{ack}, \mathsf{reset})\}$.

Proof. The proof of $\mathsf{S} = \exp(\mathsf{S}')$ is the same as [12, Proposition 2.4]. The rest of the points follow from the definition of $\mathcal{H}$. □

The behaviors of $\mathsf{val}^{\mathsf{VR}}(\theta)$ and $\mathsf{val}^{\mathsf{HR}}(\mathcal{H}(\theta))$ are related by the following notion of path stuttering (inspired by [6, Definition 7.89]):

Definition 6. *Let* $\mathsf{S} \in \mathcal{S}(\mathbb{P})$ *and* $\mathsf{S}' \in \mathcal{S}(\mathbb{P}^{\mathcal{H}})$ *be two systems and* $\mathsf{L} : \mathsf{Q}_{\mathbb{P}} \rightharpoonup \mathbb{X}$ *be a variable labeling. Given two finite or infinite firing sequences* $\rho \in \mathit{Paths}(\beta(\mathsf{S}))$ *and* $\rho' \in \mathit{Paths}(\beta(\mathsf{S}'))$*, we say that* ρ' *is a stuttering of* ρ*, and write* $\rho \trianglelefteq_{\mathsf{L}} \rho'$*, when there exists a finite or infinite sequence* $A_0, A_1, \ldots \in \mathbb{A}^{\infty}$ *and integers* $n_0, n_1, \ldots \geq 1$ *such that:*

$$\mathit{Atoms}^{\mathsf{L}\circ\lfloor 1}(\rho) = A_0, A_1, \ldots \quad \text{and} \quad \mathit{Atoms}^{\mathsf{L}^{\mathcal{H}}\circ\lfloor 1}(\rho') = \underbrace{A_0, \ldots, A_0}_{n_0 \text{ times}}, \underbrace{A_1, \ldots, A_1}_{n_1 \text{ times}}, \ldots$$

This relation is lifted to systems as $\mathsf{S} \trianglelefteq_{\mathsf{L}} \mathsf{S}'$*, if and only if the following hold:*

1. *for each* $\rho \in \mathit{Paths}(\beta(\mathsf{S}))$ *there exists* $\rho' \in \mathit{Paths}(\beta(\mathsf{S}'))$ *such that* $\rho \trianglelefteq_{\mathsf{L}} \rho'$*,*
2. *for each* $\rho' \in \mathit{Paths}(\beta(\mathsf{S}'))$ *there exists* $\rho \in \mathit{Paths}(\beta(\mathsf{S}))$ *such that* $\rho \trianglelefteq_{\mathsf{L}} \rho'$*.*

In particular $\mathsf{S} \trianglelefteq \mathsf{S}'$ implies that S and S' have sets of reachable configurations that satisfy the same atoms.

Theorem 2. *Let* $\mathsf{S} \in \mathcal{S}(\mathbb{P})$ *and* $\mathsf{S}' \in \mathcal{S}(\mathbb{P}^{\mathcal{H}})$ *be systems and* $\mathsf{L} : \mathsf{Q}_{\mathbb{P}} \rightharpoonup \mathbb{X}$ *be a variable labeling such that* $\mathsf{S} \trianglelefteq_{\mathsf{L}} \mathsf{S}'$*. Then, for any arithmetic formula* α*, we have*

$$(\mathsf{S}, \mathsf{L}) \models \alpha \iff (\mathsf{S}', \mathsf{L}^{\mathcal{H}}) \models \alpha$$

Finally, applying Theorem 2 to the lemma below (proof in Appendix 5) gives Theorem 1.

Lemma 2. *Let* θ *be a VR-term over the alphabets* $\Lambda \stackrel{\text{def}}{=} \mathbb{P} \uplus \Pi$ *and* $\Delta \stackrel{\text{def}}{=} \mathsf{T}^{obs}_{\mathbb{P}} \times \mathsf{T}^{obs}_{\mathbb{P}}$*, and* $\mathsf{L} : \mathsf{Q}_{\mathbb{P}} \to \mathbb{X}$ *be a variable labeling. Then* $\mathrm{val}^{\mathsf{VR}}(\theta) \trianglelefteq_{\mathsf{L}} \mathrm{val}^{\mathsf{HR}}(\mathcal{H}(\theta))$.

For the proof of Theorem 1, let Γ' be the HR grammar obtained by replacing each occurrence of a VR operation op from Γ by the HR operation $\mathcal{H}(\mathsf{op})$. "$\Rightarrow$" If $\mathsf{PRP}_{\mathbb{P}}(\Gamma, \alpha, \mathsf{L})$ has a positive answer, there exists a system $\mathsf{S} \in \mathcal{L}(\Gamma)$ such that $(\mathsf{S}, \mathsf{L}) \models \alpha$. Hence, there exists a ground VR-term θ such that $\mathrm{val}^{\mathsf{VR}}(\theta) = \mathsf{S}$ and θ is the result of a complete derivation of Γ. By the definition of Γ', $\mathcal{H}(\theta)$ is a ground HR-term resulting from a complete derivation of Γ'. By Lemma 2, $\mathrm{val}^{\mathsf{VR}}(\theta) \trianglelefteq_{\mathsf{L}} \mathrm{val}^{\mathsf{HR}}(\mathcal{H}(\theta))$, hence we obtain $(\mathrm{val}^{\mathsf{HR}}(\mathcal{H}(\theta)), \mathsf{L}^{\mathcal{H}}) \models \alpha$, by Theorem 2 and $\mathsf{PRP}_{\mathbb{P}^{\mathcal{H}}}(\Gamma', \phi, \mathsf{L}^{\mathcal{H}})$ has a positive answer. The "$\Leftarrow$" direction uses a symmetric argument. □

5 Conclusions and Future Work

We consider the parametric reachability problem for families of networks of finite-state processes specified using VR graph grammars. Our approach is to reduce the problem to HR grammars, for which several verification techniques have been developed lately. The reduction is based on a construction of an HR grammar that expands to the language of a given VR grammar, by elimination of epsilon-edges. We prove that this construction preserves any reachability property expressed using first-order arithmetic over variables that count the number of processes in each state.

Future work involves an implementation of the proposed reduction, based on existing tools for parametric verification of networks specified using HR grammars [9]. In particular, we plan to experiment with Azure-like network topologies and suitable routing (existence of an available route) properties. We also plan to extend the decidability result for pebble-passing systems with HR grammars [9] to VR grammars.

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. We also wish to thank the anonymous reviewers for their helpful suggestions.

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, T. Kotek, S. Rubin, F. Spegni, and H. Veith. Parameterized model checking of rendezvous systems. In P. Baldan and D. Gorla, editors, *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, volume 8704 of *Lecture Notes in Computer Science*, pages 109–124. 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. C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008.
7. 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.
8. M. Bozga and R. Iosif. Specification and safety verification of parametric hierarchical distributed systems. In G. Salaün and A. Wijs, editors, *Formal Aspects of Component Software - 17th International Conference, FACS 2021, Virtual Event, October 28-29, 2021, Proceedings*, volume 13077 of *Lecture Notes in Computer Science*, pages 95–114. Springer, 2021.
9. M. Bozga, R. Iosif, A. Sangnier, and N. Villani. Counting abstraction for the verification of structured parameterized networks. *CoRR*, abs/2502.15391, 2025.
10. M. Bozga, R. Iosif, and J. Sifakis. Verification of component-based systems with recursive architectures. *Theor. Comput. Sci.*, 940(Part):146–175, 2023.
11. M. Browne, E. Clarke, and O. Grumberg. Reasoning about networks with many identical finite state processes. *Information and Computation*, 81(1):13 – 31, 1989.
12. B. Courcelle. Structural properties of context-free sets of graphs generated by vertex replacement. *Information and Computation*, 116(2):275–293, 1995.
13. 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.
14. E. A. Emerson and K. S. Namjoshi. Reasoning about rings. In *POPL*, pages 85–94, 1995.
15. J. Engelfiet and G. Rozenberg. A comparison of boundary graph grammars and context-free hypergraph grammars. *Information and Computation*, 84(2):163–206, 1990.

16. S. M. German and A. P. Sistla. Reasoning about systems with many processes. *J. ACM*, 39(3):675–735, 1992.
17. A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: a scalable and flexible data center network. *SIGCOMM Comput. Commun. Rev.*, 39(4):51–62, Aug. 2009.
18. 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.
19. K. Jayaraman, N. S. Bjørner, J. Padhye, A. Agrawal, A. Bhargava, P. C. Bissonnette, S. Foster, A. Helwer, M. Kasten, I. Lee, A. Namdhari, H. Niaz, A. Parkhi, H. Pinnamraju, A. Power, N. M. Raje, and P. Sharma. Validating datacenters at scale. In J. Wu and W. Hall, editors, *Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM 2019, Beijing, China, August 19-23, 2019*, pages 200–213. ACM, 2019.
20. D. Le Metayer. Describing software architecture styles using graph grammars. *IEEE Transactions on Software Engineering*, 24(7):521–533, 1998.
21. 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.
22. 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.
23. 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.

Appendix

Proof of Lemma 2

In this entire section, let θ be a fixed ground VR-term written using the set $\Lambda = \mathbb{P} \uplus \Pi$ of vertex labels. It is easy to check that $\mathcal{H}(\theta)$ is a ground HR-term written using the set $\Lambda^{\mathcal{H}} \stackrel{\text{def}}{=} \mathbb{P}^{\mathcal{H}} \uplus \Pi^{\mathcal{H}}$ of vertex labels, where the sets $\mathbb{P}^{\mathcal{H}}$ of process types and $\Pi^{\mathcal{H}}$ of ports have been defined above. We write S (*resp.* S') for $\mathrm{val}^{\mathsf{VR}}(\theta)$ (*resp.* $\mathrm{val}^{\mathsf{HR}}(\mathcal{H}(\theta))$).

Take a reachable marking m of $\beta(\mathsf{S}')$, and any transition $t \in \mathsf{T}_{\mathsf{p}}^{obs}$. We say that a vertex v of type ε_t is *idle* if $\mathsf{m}(\mathsf{idle}_t, v) = 1$. We say that v of type ε_t is *waiting on* t if $\mathsf{m}(\mathsf{wait}_t, v) = 1$. We say that v of type $\mathsf{p}_{1/2}$ is *waiting* on t if $\mathsf{m}(t_{1/2}, v) = 1$. Finally, given a label $\delta \in \mathcal{E}$ and an edge $(v', \delta, v) \in \mathsf{E}_{\mathsf{S}'}$, we say that v' is an $\mathcal{E}$-predecessor of v.

Fact 1 *For any reachable marking* m *of* $\beta(\mathsf{S}')$*, a vertex* v *of type* ε_t *is non-idle in* m *if and only if there exists a* $\mathcal{E}$*-predecessor of* v *that is waiting on* t *in* m*. Furthermore if there exists such an* $\mathcal{E}$*-predecessor, there will be exactly one.*

Proof. This proof is by induction. Since the property must hold for any reachable marking, we show that it holds for $\mathrm{init}_{\beta(\mathsf{S}')}$, then that is it preserved by every firing of a transition.

In $\mathrm{init}_{\beta(\mathsf{S}')}$ all vertices of type ε_t are in their initial state and thus have a token in idle_t. All vertices of type $\mathsf{p} \in \mathbb{P}$ have a token in a place of $\mathsf{Q}_{\mathbb{P}}$, not a place from $(\mathsf{T}_{\mathbb{P}})_{1/2}$.

Therefore in the initial configuration, non-idle processes do not exist, and no process is waiting. The property is thus easy to verify.

For the inductive step, consider any transition that can occur in $\beta(\mathsf{S}')$, resulting in the firing $\mathsf{m} \overset{\delta}{\rightsquigarrow} \mathsf{m}'$. The structural properties that we established in Lemma 1 guarantee that the following case analysis is exhaustive:

- firing an internal transition t in v: no vertex of type ε_t is involved, and the vertex v in which the transition occurs is not waiting in either m or m' since ${}^\bullet t$ and $t^\bullet$ are both places of $\mathsf{Q}_\mathbb{P}$ not $(\mathsf{T}_\mathbb{P})_{1/2}$. Therefore the property holds in m'.
- firing $(v', (\mathsf{try}_t, \mathsf{recv}), v)$: for this interaction to be fireable requires that $\mathsf{m}({}^\bullet t, v') = 1$ (v' is not waiting in m) and $\mathsf{m}(\mathsf{idle}_t, v) = 1$ (v is idle in m). The inductive hypothesis applied to the latter fact yields that all $\mathcal{E}$-predecessors of v are not waiting on t in m, so when the firing makes v non-idle and v' waiting on t by $\mathsf{m}'(t_{1/2}, v') = 1$ and $\mathsf{m}'(\mathsf{active}_t, v) = 1$, it makes v' the unique $\mathcal{E}$-predecessor of v that is waiting on t in m'. Meanwhile v' is not of type ε_t so is not subject to the condition, v did not become waiting on t so the possible $\mathcal{E}$-successor of v is not affected, and all other processes retain the same state so must satisfy the same property in m' as they did in m. Therefore $\mathsf{m} \overset{(v', (\mathsf{try}_t, \mathsf{recv}), v)}{\rightsquigarrow} \mathsf{m}'$ preserves the desired property.
- firing $(v', (\mathsf{send}, \mathsf{recv}), v)$: in the same spirit as the previous case, this transition makes the token of v' move from active_t to wait_t, so v' remains non-idle and additionally becomes waiting on t in m', and at the same time the token of v moves from idle_t to active_t, so v becomes non-idle but remains non-waiting.
  Once again, v has gained exactly one waiting $\mathcal{E}$-predecessor, and all other vertices are unaffected, so m' satisfies the property.
- firing $(v', (t', t), v)$: this time we must have $\mathsf{m}(\mathsf{active}_t, v') = \mathsf{m}(\mathsf{active}_{t'}, v') = 1$, and $\mathsf{m}(\mathsf{reply}_t, v') = \mathsf{m}(\mathsf{reply}_{t'}, v') = 1$. All vertices are non-idle in m' if and only if they are non-idle in m, and they are waiting on t in m' if and only if they are waiting on t in m. The desired property is thus easily preserved.
- firing $(v', (\mathsf{commit}_t, \mathsf{reset}), v)$: this transition makes v' (waiting in m) return to non-waiting in m', and v (non-idle in m) return to idle in m'. This case is exactly symmetrical to the firing of $(v', (\mathsf{try}_t, \mathsf{recv}), v)$ and an analoguous reasoning shows that it preserves the property at hand.
- firing $(v', (\mathsf{ack}, \mathsf{reset}), v)$: similarly this case is symmetrical to $(v', (\mathsf{send}, \mathsf{recv}), v)$ and accordingly can be handled by the same reasoning.

Thus the property is preserved by all firings of transitions of $\beta(\mathsf{S}')$, and therefore holds in all reachable markings. □

We define a relation $\approx$ between markings $\mathsf{m} : \mathsf{Q}_{\beta(\mathsf{S})} \to \mathbb{N}$ and $\mathsf{m}' : \mathsf{Q}_{\beta(\mathsf{S}')} \to \mathbb{N}$, where $\mathsf{m} \approx \mathsf{m}'$ if for every place $(q, v) \in \mathsf{Q}_{\beta(\mathsf{S})}$ such that $\mathsf{m}(q, v) = 1$ we have one of the following properties:

(i) $\mathsf{m}'(q, v) = 1$, or
(ii) $\exists t \in q^\bullet.\ \exists v' \in \mathsf{V}_{\mathsf{S}'}.\ \mathsf{WaitPath}_{\mathsf{m}'}(v, v') \wedge \mathsf{m}'(t_{1/2}, v) = 1 \wedge \mathsf{m}'(\mathsf{active}_t, v') = 1$, or
(iii) $\exists t \in {}^\bullet q.\ \exists v' \in \mathsf{V}_{\mathsf{S}'}.\ \mathsf{WaitPath}_{\mathsf{m}'}(v, v') \wedge \mathsf{m}'(t_{1/2}, v) = 1 \wedge \mathsf{m}'(\mathsf{reply}_t, v') = 1$.

where we define the predicate $\mathsf{WaitPath}_{\mathsf{m}'}(v, v')$ as the existence of an $\mathcal{E}$-labeled path $v, v_1, v_2, \ldots, v_k, v'$ from v to v' such that $\mathsf{m}'(\mathsf{wait}_t, v_i) = 1$ for every $1 \leq i \leq k$. Furthermore we say that m' is *canonical* if only condition (i) is ever satisfied.

Intuitively m' is canonical and $\mathsf{m} \approx \mathsf{m}'$ when m' has tokens in the same places as m does. For non-canonical m', we can tolerate transitions that are partially executed and depending on whether the partial execution of t is less or more than half we count the token as being either in ${}^\bullet t$ or in $t^\bullet$ (recall: an $\mathcal{E}$-labeled path of S' simulates a transition of S, and the progress of this simulation is less than half if we have a path of wait_t until an active_t, and more than half if we have a path of wait_t until a reply_t).

Fact 2 *For any* $\mathsf{m} \approx \mathsf{m}'$*, we have* $Atoms^{\ell}(\mathsf{m}) = Atoms^{\ell^{\mathcal{H}}}(\mathsf{m}')$

Proof. Due to Lemma 1 and Fact 1, we know that when in the definition of $\approx$ above we talk about an $\mathcal{E}$-labeled path $\mathsf{WaitPath}_{\mathsf{m}'}(v, v')$ in the form of $v, v_1, ..., v_k, v'$, it necessarily means that all $v_1, ..., v_k$ are vertices of type ε_t, and that they are disjoint from any other instanciation of $\mathsf{WaitPath}_{\mathsf{m}'}$ for a different place. In particular this implies that the conditions (i), (ii), (iii) are mutually exclusive, and any token in a place active_t or reply_t can be injectively mapped to some vertex v with a token in $t_{1/2}$ from which there is a $\mathsf{WaitPath}_{\mathsf{m}'}$ as described.

This induces a bijection between $\{v \in \mathsf{V_S} \mid \mathsf{m}(q, v) = 1\}$, and $\{v \in \mathsf{V_{S'}} \mid \mathsf{m}'(q, v) = 1\} \uplus \{v \in \mathsf{V_{S'}} \mid \mathsf{m}'(\mathsf{active}_t, v) = 1 \text{ for } t \in q^\bullet\} \uplus \{v \in \mathsf{V_{S'}} \mid \mathsf{m}'(\mathsf{reply}_t, v) = 1 \text{ for } t \in {}^\bullet q\}$, implying that they have the same cardinal. It happens that $\sum_{q \in \ell^{-1}(x)} \mathsf{m}(q)$ is exactly the cardinal of the first set, and $\sum_{q \in (\ell^{\mathcal{H}})^{-1}(x)} \mathsf{m}'(q)$ is the cardinal of the second. We conclude that ℓ and $\ell^{\mathcal{H}}$ have the same interpretation of x, and since this applies to every $x \in \mathbb{X}$ we conclude that the same atoms are satisfied in m and in m'. □

Fact 3 *If* $\mathsf{m}'_0 \overset{\delta}{\rightsquigarrow} \mathsf{m}'_1$ *in* $\beta(\mathsf{S}')$ *and* $\delta \in \mathcal{E} \cup \overleftarrow{\mathcal{E}}$*, then for all* m *marking of* $\beta(\mathsf{S})$ *we have* $\mathsf{m} \approx \mathsf{m}'_0$ *if and only if* $\mathsf{m} \approx \mathsf{m}'_1$.

Proof. Once again we rely on Lemma 1 and Fact 1. Consider an interaction that can be fired between some v and v', and assume that it results in the transition from m'_0 to m'_1. Since $\approx$ is defined pointwise on each place of $\beta(\mathsf{S})$ that holds a token, and since m is unchanged, it suffices to look at exactly the places of $\beta(\mathsf{S}')$ that are changed by $\mathsf{m}'_0 \overset{\delta}{\rightsquigarrow} \mathsf{m}'_1$. In all configurations we will show that the equivalence is preserved simply by changing which of the conditions (i), (ii), (iii) hold on each place. In all that follows we will write t the transition that we consider (fully determined by the type ε_t of v'), and $v_0 \in \mathsf{V_S}$ the vertex that is currently being considered for the equivalence. The cases are as follows:

- $(v, (\mathsf{try}_t, \mathsf{recv}), v')$: m is linked to m'_0 on $({}^\bullet t, v)$ by condition (i). Furthermore $v_0 = v$. We show that after firing $\delta = (\mathsf{try}_t, \mathsf{recv})$ the markings are now linked by condition (ii) on the path v, v'.
  In m'_0, the vertex v which is of type $\mathsf{p}_{1/2}$ has a token in place ${}^\bullet t$. We know this because it is the only way that its transition try_t is fireable, and this is why condition (i) is satisfied. Similarly there must be a token in (idle_t, v') so that recv is fireable.
  After firing δ we now have $\mathsf{m}'_1(t_{1/2}, v) = 1$ and $\mathsf{m}'_1(\mathsf{active}_t, v') = 1$. This satisfies the condition for (ii) with $\mathsf{WaitPath}_{\mathsf{m}'_1}(v, v')$ instanciated by the trivial path v, v'. Every step of this reasoning is in fact an equivalence, so we also get the other direction of the statement.

– $(v,(\mathsf{send},\mathsf{recv}),v')$: m and m'_0 are linked on $({}^\bullet t, v_0)$ by condition (ii), in the form of $\mathsf{WaitPath}_{\mathsf{m}'_0}(v_0,v)$ instanciated by some $v_0,v_1,...,v_k,v$. We show that m and m'_1 are still linked by condition (ii), but on v' instead of v.
The interaction $(\mathsf{send},\mathsf{recv})$ belongs to $\mathcal{E}$, so $v_0,v_1,...,v_k,v,v'$ is still an $\mathcal{E}$-labeled path. In fact after we fire δ, we get $\mathsf{m}'_1(\mathsf{wait}_t,v)=1$ which means that $v_0,v_1,...,v_k,v,v'$ instanciates $\mathsf{WaitPath}_{\mathsf{m}'}(v_0,v')$. Lastly we have $\mathsf{m}'_1(\mathsf{active}_t,v')=1$, thus satisfying condition (ii).
– $(v,(\mathsf{commit}_t,\mathsf{reset}),v')$: m is linked to m'_0 on $(t^\bullet,v_0)$ by condition (iii). With a reasoning analoguous to the first case, including the fact that $v_0=v$, we can prove that m and m'_1 are instead linked by condition (i).
– $(v,(\mathsf{ack},\mathsf{reset}),v')$: m is linked to m'_0 on $(t^\bullet,v_0)$ by condition (iii). With a reasoning analoguous to the second case, we can prove that m and m'_1 are still linked by condition (iii), but using a shorter $\mathsf{WaitPath}(v_0,v)$ instead of $\mathsf{WaitPath}(v_0,v')$.

□

Fact 4 *For any markings* $\mathsf{m}_1,\mathsf{m}_2:\mathsf{Q}_{\beta(\mathsf{S})}\to\mathbb{N}$*, transition* $\delta\in\mathsf{T}_{\beta(\mathsf{S})}$ *such that* $\mathsf{m}_1\overset{\delta}{\rightsquigarrow}\mathsf{m}_2$ *in* $\beta(\mathsf{S})$*, and any* $\mathsf{m}'_1:\mathsf{Q}_{\beta(\mathsf{S}')}\to\mathbb{N}$*, if* $\mathsf{m}_1\approx\mathsf{m}'_1$ *and* m'_1 *is canonical then there exist a marking* $\mathsf{m}'_2:\mathsf{Q}_{\beta(\mathsf{S}')}\to\mathbb{N}$*, an* $\mathcal{E}$*-sequence* $\mathbf{p}$*, and an* $\overleftarrow{\mathcal{E}}$*-sequence* $\mathbf{q}$*, such that* $\mathsf{m}'_1\overset{\mathbf{p}\cdot\delta\cdot\mathbf{q}}{\rightsquigarrow}\mathsf{m}'_2$ *in* $\beta(\mathsf{S}')$ *and* $\mathsf{m}_2\approx\mathsf{m}'_2$ *with* m'_2 *canonical.*

Proof. Firstly if δ is an internal transition, then the canonicity of $\mathsf{m}_1\approx\mathsf{m}_2$ and the fact that process types in $\mathbb{P}_{1/2}$ have the same internal transitions as those of $\mathbb{P}$, give that δ must also be fireable in m_2. We thus choose both $\mathbf{p}$ and $\mathbf{q}$ to be the empty sequence.

Otherwise, we assume that from m_1 to m_2, the interaction $\delta=(t,t')$ occurs between vertices v and v'. As we hinted at in Example 8, we choose $\mathbf{p}$ to be $(\mathsf{try}_t,\mathsf{recv}),(\mathsf{send},\mathsf{recv}),...,(\mathsf{send},\mathsf{recv})$ on an ε_t-path starting from v, followed by $(\mathsf{try}_{t'},\mathsf{recv}),(\mathsf{send},\mathsf{recv}),...,(\mathsf{send},\mathsf{recv})$, on an $\varepsilon_{t'}$-path starting from v'. After that we execute (t,t'), then propagate backwards for $\mathbf{q}$ as $(\mathsf{reset},\mathsf{ack}),...,(\mathsf{reset},\mathsf{ack}),(\mathsf{reset},\mathsf{commit}_t)$ and $(\mathsf{reset},\mathsf{ack}),...,(\mathsf{reset},\mathsf{ack}),(\mathsf{reset},\mathsf{commit}_{t'})$ along the same two paths in reverse.

Since m'_1 is canonical, all vertices of type ε_t and $\varepsilon_{t'}$ in S' have a token in idle_t in m'_1. Since $\mathsf{S}=\exp(\mathsf{S}')$, the existence of an interaction (t,t') between v and v' implies that there are some u,u' in S' and there exist an ε_t-path from v to u and an $\varepsilon_{t'}$-path from v' to u'. Along such paths, the sequence of transitions described above is a firing sequence if initially all vertices of type ε_t and $\varepsilon_{t'}$ are in state idle_t and $\mathsf{idle}_{t'}$ respectively, and they will return to state idle_t or $\mathsf{idle}_{t'}$ once the entire sequence is fired.

Thus $\mathsf{m}'_1\overset{\mathbf{p}\cdot\delta\cdot\mathbf{q}}{\rightsquigarrow}\mathsf{m}'_2$, m'_2 is canonical, and $\mathsf{m}_2\approx\mathsf{m}'_2$. □

Fact 5 *Every (finite or infinite) firing sequence in* $\mathit{Paths}(\beta(\mathsf{S}))$ *admits a stuttering firing sequence in* $\mathit{Paths}(\beta(\mathsf{S}'))$.

Proof. Let $\mathrm{init}_{\beta(\mathsf{S})}=\mathsf{m}_0\overset{\delta_0}{\rightsquigarrow}\mathsf{m}_1\overset{\delta_1}{\rightsquigarrow}\mathsf{m}_2\overset{\delta_2}{\rightsquigarrow}\cdots$ be a (finite or infinite) firing sequence $\rho\in\mathit{Paths}(\beta(\mathsf{S}))$. Using the initialization that $\mathrm{init}_{\beta(\mathsf{S})}\approx\mathrm{init}_{\beta(\mathsf{S}')}$, and the latter is canonical (since the initial token of ε_t is in idle_t for all t), we can inductively apply Fact 4 at every step to construct a firing sequence $\mathrm{init}_{\beta(\mathsf{S}')}=\mathsf{m}'_0\overset{\mathbf{p}_0\cdot\delta_0\mathbf{q}_0}{\rightsquigarrow}\mathsf{m}'_1\overset{\mathbf{p}_1\cdot\delta_1\cdot\mathbf{q}_1}{\rightsquigarrow}\mathsf{m}'_2\overset{\mathbf{p}_2\cdot\delta_2\cdot\mathbf{q}_2}{\rightsquigarrow}\cdots$ which

is a path $\rho' \in \mathit{Paths}(\beta(S'))$. In this construction all $\mathbf{p}_i$ and $\mathbf{q}_i$ are $\mathcal{E}$-sequences and $\overleftarrow{\mathcal{E}}$-sequences respectively, thus invariant for $\approx$ (Fact 3). This in turn implies that they do not change the set of atomic propositions that hold (Fact 2).

A path $\mathit{Atoms}^{\ell}(\rho) = A_0A_1A_2\cdots$, where $\rho \in \mathit{Paths}(\beta(S))$ is thus transformed by this construction into a path $\mathit{Atoms}^{\ell^{\mathcal{H}}}(\rho') = A_0^{1+|\mathbf{p}_0|}A_1^{1+|\mathbf{q}_0|+|\mathbf{p}_1|}A_2^{1+|\mathbf{q}_1|+|\mathbf{p}_2|}\cdots$ which gives $\rho \trianglelefteq_{\mathsf{L}} \rho'$, for some $\rho' \in \mathit{Paths}(\beta(S'))$. □

Fact 6 *Any firing sequence of $\mathcal{E} \cup \overleftarrow{\mathcal{E}}$-transitions must be finite.*

Proof. In principle, the proof is straightforward: we define a cost function on places and show that every $\mathcal{E}$-labeled transition moves a token to a place that has a strictly smaller cost. Let $c : \mathsf{Q}_{\beta(S')} \to \mathbb{N}$ be defined as follows:

$$
\begin{aligned}
c(\mathsf{wait}_t, v) = c(\mathsf{idle}_t, v) = c(t_{1/2}, v) &= 0 && \text{for any } t, v\\
c(\mathsf{active}_t, v) &= 1 + c(\mathsf{active}_t, v') && \text{if } \exists v'.\ v \xrightarrow{\mathcal{E}} v'\\
c(\mathsf{active}_t, v) &= 0 && \text{otherwise}\\
c(q, v) &= 1 + \max_{t \in q^\bullet} \max_{v' : v \xrightarrow{\mathcal{E}} v'} c(\mathsf{active}_t, v') && \text{for any } q, v\\
c(\mathsf{reply}_t, v) &= 1 + \max_{v' : v' \xrightarrow{\mathcal{E}} v} c(t^\bullet, v) + \max_{v' : v' \xrightarrow{\mathcal{E}} v} c(\mathsf{reply}_t, v') && \text{for any } t, v
\end{aligned}
$$

This function, apart from being obviously nonnegative, is well-defined everywhere: the acyclicity of the subgraph of S' composed only of $\mathcal{E}$-edges guarantees that the system of equations above has a solution. More precisely:

- c is obviously well-defined for places of the form (wait_t, v), (idle_t, v), or $(t_{1/2}, v)$;
- when a token is in a place (active_t, v) it can only be involved in transitions in the direction of $\mathcal{E}$ which is acyclic. Furthermore vertices ε_t form a forest so if v' such that $v \xrightarrow{\mathcal{E}} v'$ exists then it is unique, thus c is also well-defined for places of the form (active_t, v);
- finally c over (q, v) and (reply_t, v) is defined in terms of $\mathcal{E}$-predecessors, which are not unique (thus why we need a max), but still form an acyclic graph.

Given a marking m, we define

$$\mathrm{fuel}(\mathsf{m}) \stackrel{\mathsf{def}}{=} \sum_{q \in \mathsf{Q}_{S'}} \mathsf{m}(q)c(q)$$

Observe that since $\mathsf{m}(q)$ can only take the value 0 or 1, the fuel of a marking is the sum of the cost of overy place with a token. Our objective is now to show that whenever $\mathsf{m} \overset{t_0}{\rightsquigarrow} \mathsf{m}'$ for $t_0 \in \mathsf{T}_{\beta(S')}$ labeled by $\delta \in \mathcal{E} \cup \overleftarrow{\mathcal{E}}$, we have $\mathrm{fuel}(\mathsf{m}) > \mathrm{fuel}(\mathsf{m}')$. We do this by showing the equivalent fact that $\mathrm{fuel}(\mathsf{m}) - \mathrm{fuel}(\mathsf{m}') > 0$ because most of it cancels out: partition the set of vertices into ${}^\bullet t_0$ (loses a token when we fire t_0), $t_0{}^\bullet$ (receives a token when we fire t_0), and the rest (unchanged), and we are left with simply

$$\mathrm{fuel}(\mathsf{m}) - \mathrm{fuel}(\mathsf{m}') = \sum_{q \in {}^\bullet t} c(q) - \sum_{q \in t^\bullet} c(q)$$

Now for the case analysis on δ:

- $(v,(\mathsf{try}_t,\mathsf{recv}),v')$: $\mathsf{fuel}(\mathsf{m})-\mathsf{fuel}(\mathsf{m}')=c({}^\bullet t,v)+c(\mathsf{idle}_t,v')-c(t_{1/2},v)-c(\mathsf{active}_t,v')=c({}^\bullet t,v)-c(\mathsf{active}_t,v')$. Since $v\xrightarrow{\mathcal{E}}v'$, (active_t,v') is already accounted for in $c({}^\bullet t,v)=1+\max_{t\in({}^\bullet t)^\bullet}\max_{v':v\xrightarrow{\mathcal{E}}v'}c(\mathsf{active}_t,v')$, and thus $c({}^\bullet t,v)>c(\mathsf{active}_t,v')$. We easily conclude that $\mathsf{fuel}(\mathsf{m})-\mathsf{fuel}(\mathsf{m}')>0$.
- $(v,(\mathsf{send},\mathsf{recv}),v')$: $\mathsf{fuel}(\mathsf{m})-\mathsf{fuel}(\mathsf{m}')=c(\mathsf{active}_t,v)+c(\mathsf{idle}_t,v')-c(\mathsf{idle}_t,v)-c(\mathsf{active}_t,v')=c(\mathsf{active}_t,v)-c(\mathsf{active}_t,v')=(1+c(\mathsf{active}_t,v'))-c(\mathsf{active}_t,v')=1$ strictly positive as expected.
- $(v,(\mathsf{commit}_t,\mathsf{reset}),v')$: $\mathsf{fuel}(\mathsf{m})-\mathsf{fuel}(\mathsf{m}')=c(t_{1/2},v)+c(\mathsf{reply}_t,v')-c(t^\bullet,v)-c(\mathsf{idle}_t,v')=c(\mathsf{reply}_t,v')-c(t^\bullet,v)$. Since the quantity $c(\mathsf{reply}_t,v)$ occurs in $\max_{v:v\xrightarrow{\mathcal{E}}v'}c(\mathsf{reply}_t,v)$ which is part of the sum that constructs $c(\mathsf{reply}_t,v')$, the inequality is obvious.
- $(v,(\mathsf{ack},\mathsf{reset}),v')$: The same reasoning as the previous case applies, since this time $\mathsf{fuel}(\mathsf{m})-\mathsf{fuel}(\mathsf{m}')=c(\mathsf{reply}_t,v)-c(\mathsf{reply}_t,v')$ and once again $c(\mathsf{reply}_t,v')$ occurs in $\max_{v:v\xrightarrow{\mathcal{E}}v'}c(\mathsf{reply}_t,v)$ leading to $c(\mathsf{reply}_t,v)-c(\mathsf{reply}_t,v')>0$.

In all cases we obtain $\mathsf{fuel}(\mathsf{m})>\mathsf{fuel}(\mathsf{m}')$. The function fuel, as we have already justified, is a well-defined finite nonnegative quantity. It follows that an infinite firing sequence of $\mathcal{E}\cup\overleftarrow{\mathcal{E}}$-labeled edges cannot exist. □

Fact 7 *For any markings* $\mathsf{m}'_1,\mathsf{m}'_2:\mathsf{Q}_{\beta(\mathsf{S}')}\to\mathbb{N}$, *transition* $\delta\in\mathsf{T}_{\beta(\mathsf{S}')}$ *such that* $\mathsf{m}'_1\overset{\delta}{\rightsquigarrow}\mathsf{m}'_2$ *in* $\beta(\mathsf{S})$, *and any* $\mathsf{m}_1:\mathsf{Q}_{\beta(\mathsf{S})}\to\mathbb{N}$, *if* $\mathsf{m}_1\approx\mathsf{m}'_1$ *and* $\delta\notin\mathcal{E}\cup\overleftarrow{\mathcal{E}}$ *then there exists a marking* $\mathsf{m}_2:\mathsf{Q}_{\beta(\mathsf{S})}\to\mathbb{N}$ *such that* $\mathsf{m}_1\overset{\delta}{\rightsquigarrow}\mathsf{m}_2$ *in* $\beta(\mathsf{S})$ *and* $\mathsf{m}_2\approx\mathsf{m}'_2$.

Proof. If $\delta=t$ is an internal transition, it means that for some $(q,v)={}^\bullet t$ we have $\mathsf{m}'_1(q,v)=1$. Thee definition of $\approx$ this implies $\mathsf{m}_1(q,v)=1$ and thus δ is also fireable in $\beta(\mathsf{S})$ and obviously leads to $\mathsf{m}_2\approx\mathsf{m}'_2$.

Otherwise $\delta=(t,t')$ is an observable transition. It must occur between some e and e' of respective types ε_t and $\varepsilon_{t'}$, and in m_2 they must have a token in active_t and $\mathsf{active}_{t'}$ respectively. By the characterization from Fact 1, there must furthermore exist some vertices v,v' such that there is an ε_t-path from v to e and an $\varepsilon_{t'}$-path from v' to e', and all vertices along those paths have a token in wait_t or $\mathsf{wait}_{t'}$ respectively. Finally, still from Fact 1, we obtain that in m'_1, v and v' have a token in $t_{1/2}$ and $t'_{1/2}$ respectively.

Knowing that $\mathsf{m}_1\approx\mathsf{m}'_1$, this means that $\mathsf{m}_1({}^\bullet t,v)=1$ and $\mathsf{m}_1({}^\bullet t',v)=1$, and thus (t,t') is fireable in m'_1. Firing it yields m'_2 with $\mathsf{m}_2(t^\bullet,v)=1$ and $\mathsf{m}_2(t'^\bullet,v)=1$. On the other hand m'_2 only differs from m'_1 in that instead of having $\mathsf{m}'_1(\mathsf{active}_t,e)=1$ and $\mathsf{m}'_1(\mathsf{active}_{t'},e')=1$ we now have $\mathsf{m}'_2(\mathsf{reply}_t,e)=1$ and $\mathsf{m}'_2(\mathsf{reply}_{t'},e')=1$. In the definition of $\approx$, we thus easily check that $\mathsf{m}_2\approx\mathsf{m}'_2$. □

Fact 8 *Every (finite or infinite) firing sequence in* $\mathit{Paths}(\beta(\mathsf{S}'))$ *is a stuttering of a firing sequence in* $\mathit{Paths}(\beta(\mathsf{S}))$.

Proof. Let $\mathrm{init}_{\beta(\mathsf{S}')}=\mathsf{m}'_0\overset{t_1}{\rightsquigarrow}\mathsf{m}'_1\overset{t_2}{\rightsquigarrow}\cdots$ be a firing sequence $\rho'\in\mathit{Paths}(\beta(\mathsf{S}'))$. Let $i_0=0$ then $i_1,i_2,\ldots$ be an enumeration of all the indices i for which $t_i\notin\mathcal{E}\cup\overleftarrow{\mathcal{E}}$. Starting from $\mathrm{init}_{\beta(\mathsf{S}')}=\mathsf{m}_0$, we apply Fact 7 for every i_j in increasing order, to get a firing sequence ρ defined by $\mathsf{m}_0\overset{t_{i_1}}{\rightsquigarrow}\mathsf{m}_1\overset{t_{i_2}}{\rightsquigarrow}\cdots$ and such that $\mathsf{m}_j\approx\mathsf{m}'_{i_j}$ for every $j\geq 0$. Applying Fact 3

in-between gives that $\mathsf{m}_j \approx \mathsf{m}'_k$ for every $i_j \leq k < i_{j+1}$, and Fact 6 guarantees that there are finitely many stuttering steps. We can finally conclude $\mathit{Atoms}^{\ell}(\rho) \trianglelefteq \mathit{Atoms}^{\ell^{\mathcal{H}}}(\rho')$. □

Finally we conclude from Fact 5 and Fact 8 that $\beta(\mathsf{S}) \trianglelefteq_{\mathsf{L}} \beta(\mathsf{S}')$: $\beta(\mathsf{S})$ and $\beta(\mathsf{S}')$ have the same set of paths, up to finite stuttering on the side of S'.